HyperCoin Workshop III

November 30 - December 1, 1996
Ritz-Carlton Hotel, San Francisco

Participants

Mr Dennis Allison
514 Bryant Street
Palo Alto, CA
94301 USA

Phone: + 1 415 856-8720 (home)
+ 1 415 723-9213 (office)
Fax: + 1 415 725-7398
Email: allison@sumeru.Stanford.edu

Mr Jock Gill
18 Woods Edge Road
Medford, MA 02155-2316
Phone: + 1 617 396-0492
Fax:
Email: jgill@penfield-gill.com
Web: www.penfield-gill.com/gill

Mr Stuart Haber

Phone: + 1 201 701-8192
Fax: + 1 201 701-0601
Email: stuart@surety.com

Mr Shin Kumada
ASCII
Tohshin Building
4-33-10 Yoyogi Shibuya-ku
Tokyo 151-24 Japan

Phone: + 81 3 5351-8455
Fax: + 81 3 5351-8465
Email: shi-ku@ascii.co.jp

Mr Masato Kunii

Phone: + 81 3 5322-3400
Fax: + 81 3 5322-3401
Email: pro009y@auth.proside.co.jp

Mr David Lytel
1722 V Street, NW
Washington, DC 20009

Phone: + 1 202 588-7400
Fax: + 1 202 319-8330
Email: david@lytel.com

Ms Marlene Mallicoat
(Permanent:)
3020 Bridgeway #295
Sausalito, CA 94965

Phone: + 1 415 331-4422
Fax: + 1 415 332-0136
Email: marlene@xanadu.net

(Temporary until at least 97.4.1:)
Royal Emi Shonan #402
2-18-9 Shonandai
Fujisawa, 252, Japan

Phone: + 81 466 46 7368
Fax: (same)
Email: marlene@xanadu.net

Mr Ted Nelson
(Permanent:)
3020 Bridgeway #295
Sausalito, CA 94965

Phone: + 1 415 331-4422
Fax: + 1 415 332-0136
Email: ted@xanadu.net

(Temporary until at least 97.4.1:)
Royal Emi Shonan #402
2-18-9 Shonandai
Fujisawa, 252, Japan
Phone: + 81 466 46 7368
Fax: (same)
Email: ted@xanadu.net

Mr Kay Nishi
ASCII Corporation
Tohshin Building
4-33-10 Yoyogi Shibuya-ku
Tokyo 151-24 Japan

Phone: + 81 3 5351-8455
Fax: + 81 3 5351-8465
Email: nishi@ascii.co.jp

Administative Assistants:
Ms. Kazamaki kazamaki@ascii.co.jp
Mr Kumada shi-ku@ascii.co.jp

Mr Andrew Pam
PO Box 409
Canterbury
VIC 3126 Australia

Phone: + 61 3 9419 2989 (home)
Phone: + 61 3 9651-1511 (office)
Fax: + 61 3 9651-1502
Email: avatar@glasswings.com.au

Mr Koji Shimizu

Phone: + 81 3 5322 3400
Fax: + 81 3 5322 3401
Email: pro009y@auth.proside.co.jp

Mr Tsutomu Shimomura
UC San Diego
SDSC 0505
La Jolla, CA 92093-0505 USA

Phone: + 1 619 534-5050
Fax: + 1 619 534 5152
Email: tsutomu@ariel.sdsc.edu

Mr Dave Theurer
10300 Mira Vista Road
Cupertino, CA 95014 USA

Phone: + 1 408 725-8849
Fax: + 1 408 725 8855
Email: theurer@netcom.com

Mr Kevin Vaughn
HarperCollins
1160 Battery Street
San Francisco, CA 94111

Phone: + 1 415 477-4419
Fax: + 1 415 477-4444
Email: kevin.vaughn@harpercollins.com

Mr Jared Vogt
MetaInfo, Inc
700 5th Avenue, Suite 5500
Seattle, WA 98104

Phone: + 1 206 523-0484
Fax: + 1 206 467-9237
Email: jaredv@metainfo.com

Mr Philip Zimmermann
(Not in attendance at this workshop)
PGP
555 Twin Dolphin Drive, Suite 570
Redwood Shores, CA 94065

Phone: + 1 415 631 1747
Fax: + 1 415 631 0599
Email: prz@pgp.com
Web: ...www.pgp.com

*****

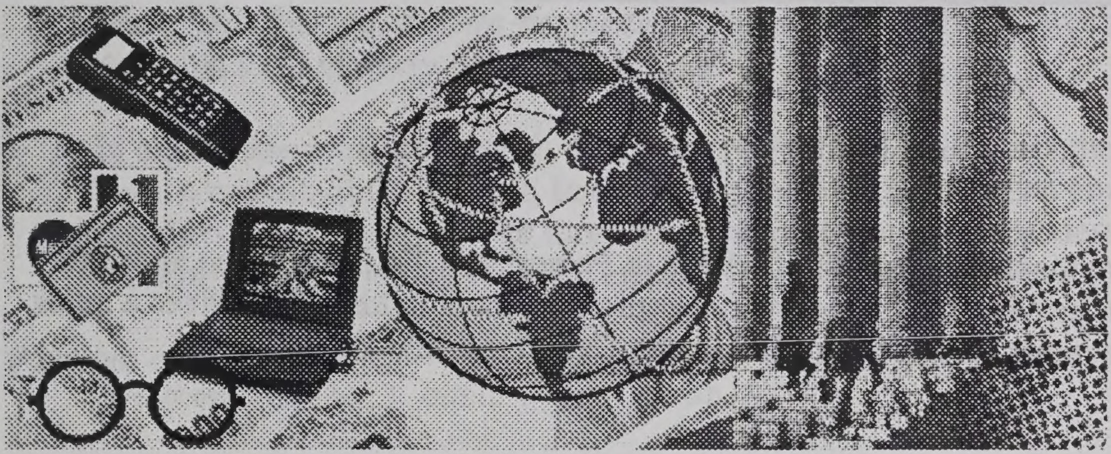
HyperCoin Web Site: http://hypercoin.net:8080/hypercoin

http://www.research.digital.com/
SRC/millicent

digital

MILLICENT

Digital's Microcommerce System



- **What's New**
- **Technology**
- **Whitepapers**
- **Presentations**
- **Specifications**
- **Questions**
- **Other Sites**
- **Glossary**
- **Contacts**

Millicent (patent pending) is a lightweight and secure protocol for electronic commerce over the Internet. As a microcommerce system, Millicent is designed to support purchases costing fractions of a cent.

The name *Millicent* is derived from the underlying cost associated with each micropayment transaction. Millicent is based on decentralized validation of electronic cash at the content providers Web server without any additional communication, expensive encryption, or off-line processing. ← Specs Attached

The key innovations of Millicent are its use of brokers and of scrip. Brokers take care of account management, billing, connection maintenance, and establishing accounts with vendors. Scrip is microcurrency that is only valid within the Millicent-enabled world.

If you would like to be kept abreast on Millicent development, and added to our Millicent mailing list, please send email to millicent@digital.com.

digital

MILLICENT

Digital's Microcommerce System

What's New

October 1996
Progress Report on Micropayments presentation (399KB in PowerPoint V4.0) has been added to the server. This presentation provides an overview to the micropayment industry and gives a quick snapshot of current micropayment efforts from ClickShare, CyberCash and Digital.

September 1996
Millicent Protocol for Inexpensive Electronic Commerce presentation (240KB in PowerPoint V4.0) has been revised for the DECUS'96 Conference.

August 1996
Millicent is featured in a Wired Magazine article on micropayment technology, Nanobucks: Big deal or pocket change? Good profile of all of the micropayment schemes.

May 1996
Microcommerce Overview presentation (421KB in PowerPoint V4.0) has been added to the server. This presentation serves as an introduction to micropayment technology and its different applications.

January 1996
Business Week (1/15/96) covers Millicent in an article called *Cyberscrip: When A Penny is Too Much To Pay*. Bob Metcalfe devotes his January 29th, 1996 InfoWorld column to Millicent.

December 1995
Millicent Protocol for Inexpensive Electronic Commerce presentation (71KB in PowerPoint V4.0) presented at the 4th International World Wide Web Conference.

TM

An HTTP protocol for payment

Common fields

Payment protocol type

```
"Payment-Protocol:" protocol-name ["," protocol-name]*
```

There must be one or more *Payment-Protocol* fields in every payment request and error reply, and one such field in every normal reply.

In a purchase request or reply, the Payment-Protocol field gives the protocol that is used, or has been accepted, in the request. In an *inquiry* request, in which an empty bid is made (see below), the client may specify one or more payment protocols that it undestands. In an error response the meaning varies with the returned error.

Payment cash

```
"Payment-Cash: " cash [ ", " cash ]*
```

There may be any number of *Payment-Cash* fields in a request, normal reply, or error reply.

In a request, these fields specify pieces of electronic currency that may be used to pay for the requested item. A purchase request uses only one payment protocol, specified in the Payment-Protocol field; all the Payment-Cash fields are assumed to have been defined in that protocol.

In a normal reply, this field will return the scrip from the purchase. This will include the change (if any) from the purchase and, for a broker, any scrip purchased as a result of the request.

In an error reply in which the error arose from a problem with the cash submitted for payment, this field will return the piece of cash that generated the error. Such replies will define only one payment protocol in the Payment-Protocol field; this defines the protocol for the returned Payment-Cash field.

Payment authorisation

```
"Payment-Authorization: " authorization
```

There must be at most one *Payment-Authorisation* field in a request or reply. The use of the authorisation field depends on the payment protocol

In a purchase request this field certifies that the request came from the customer. In a normal purchase reply, this field may certify that any returned change and/or the body of the reply came from the vendor.

Client to vendor

Payment bid

```
"Payment-Bid:" [bid-info [";"guarantee]]
```

There is at most one *Payment-Bid* in a request.

In a request, this field allows the vendor to charge up to the amount specified in the *bid-info* field. If the *bid-info* field is present, it is is assumed to refer to the protocol defined in the Payment-Protocol field. If there is a guarantee, then the customer believes that the price in *bid-info* has been guaranteed by the vendor.

The vendor may reject the bid price and return an error reply with Payment-Offer fields. A customer can get a price from a vendor by sending a bid message with no bid-info information and catching the returned error message. In this case, the customer may send multiple payment protocols as a hint to the vendor of which protocols to provide offers for.

Vendor to client

Payment receipt

```
"Payment-Receipt:" price-info [";" receipt-info ]
```

There is at most one *Payment-Receipt* field in a normal reply to a purchase message.

In a reply, this field gives the price of the purchased item and shows that cash has been used to pay for it. It may also include a *receipt-info* field that proves to the vendor that the item has been purchased by this customer.

Payment offer

```
"Payment-Offer:" protocol-name "/" offer-info ["/" guarantee] [","
protocol-name "/" offer-info ["/" guarantee ]]*
```

There may be any number of *Payment-Offer* fields in an error reply.

In an error reply, this field indicates the minimum price of the unpurchased item. The vendor may also return a guarantee that the price will be honored until a given date and time. The client should send the guarantee with the relevant Payement-Bid message. It is up to the vendor to decide how strong the guarantee is.

Payment error

```
"Payment-Error:" error-description
```

There must at most one *Payment-Error* field in an error reply. There must be a payment error field in an error reply where the error status is `420 Bad Payment` or `422 Payment Rejected`

Error results

The following error types may be returned by any payment system.

```
HTTP/1.0 402 Insufficient Payment
```

The request include insufficient payment, possibly none, for the item. The reply must include a Payment-Protocol field and at least one Payment-Offer field.

`HTTP/1.0 420 Bad Payment`

There was an error in the payment. The reply must include a Payment-Protocol field and at least one Payment-Error field.

`HTTP/1.0 421 Unknown Payment Protocol`

The protocol described in the Payment-Protocol field is not supported by the vendor. The reply must include one or more Payment-Protocol fields that list the supported protocols. It may also include at most one Payment-Offer field for each supported protocol.

`HTTP/1.0 422 Payment Rejected`

The payment has been rejected for a reason other than a failure in the payment system. No other Payment fields are required in the reply.

Reserved URLs

Customers, vendors and brokers will sometimes need to communicate for other reasons than making a purchase. For example, a customer may need to break a single large piece of digital cash into multiple smaller pieces to allow multiple simultaneous purchases. To support this, we reserve a part of a server's URL space for each supported payment protocol:

```
"http://"server"/payment.000/"payment-protocol
```

Each payment protocol may then use addresses within its URL space to provide access to its "meta-transactions", i.e. transactions about transactions.

In addition, a customer can find the protocols that a server supports by retrieving the page at `"http://"server"/payment.000/`

Protocol-specific details

The formats of *authorization*, *bid-info*, *cash*, *error-description*, *guarantee*, *offer-info*, *price-info* and *receipt-info* are protcol-specific.

To simplify high-level parsing, protocol names and protocol-specific encodings of cash, offer-info and guarantee must not use the comma (,) character

The only specific protocol defined so far is Millicent

$Id: payproto.htm,v 1.3 1995/11/06 20:50:49 sfreeman Exp sfreeman $

Millicent-specific elements for an HTTP payment protocol

This page describes the sub-protocol required to use Millicent with the HTTP payment protocol.

The string to identify the current Millicent protocol is: `Millicent-1.0`

Top-level field types

This section describes the layout of the fields for the Millicent protocol.

authorization

The authorization is the signature of a message to validate its source. The format is: `["body: body-checksum;"]signature`, where *body-checksum* is a hash of the HTML body of the message and *signature* is a hash of the concatenation, without spaces, of the authorization fields.

For a request, the hash key is the governing secret and the authorization fields are (in order with no intervening characters):

- the server-relative portion of the URL; and
- if the *body* keyword is present, the body-checksum of the request; and
- all the Millicent cash strings in ascending ascii order of vendor name concatenated with the scrip-id; and

Authorisations are optional for normal replies. If present, the authorization fields are:

- the server-relative portion of the URL; and
- the contents of the payment receipt field; and
- if present, the contents of the cash fields (ie. returned change and purchased scrip) in ascending ascii order of vendor name concatenated with the scrip-id; and
- if the *body* keyword is present, the body-checksum of the returned item; and,

bid-info

The bid-info describes the largest amount the customer is willing to pay for a requested service. The format of a bid-info is:

```
price | [currency-code]
```

A customer may make a firm bid, which will include a *price* field. Otherwise, the customer may optionally send a *currency-code*, as defined in the format for prices, as a hint to the vendor of what type of price information to return.

cash

A cash field is a scrip, a piece of digital currency. Its format is:

```
vendor "/" value "/" scrip-id "/" customer-id "/" expiration-date [ "{"
prop-name "=" prop-value ["/" prop-name "=" prop-value]* "}"] "<"
vendor-signature ">" [ "<"new-customer-secret">"]
```

where *scrip-id* and *customer-id* are integers, and *prop-name* and *prop-value* are strings made up of any printable character except those in "`{}=,/+<>`". The only permissible use of whitespace is in *prop-value*s enclosed in quotations. *vendor-signature* is a opaque text sent by the vendor to certify the scrip.

In a reply, a vendor may also send a *new-customer-secret* which is to be associated with the returned cash.

error-description

An error description is a comma-separated list of error values. The error reply must also include the field that caused the error to be returned, which implies that an error is returned as soon as the first problem is discovered. The known error values are:

- *Invalid_scrip*: The scrip given in payment was invalid.
- *Expired_scrip*: The scrip given in payment has expired.
- *Double_spent_scrip*: The scrip given in payment has already expired.
- *Bad_properties*: The properties of the scrip do not allow the purchase.
- *Bad_request_signature*: The request signature is invalid.
- *Bad_currency*: The server cannot support the requested currency.
- *Too_many_pieces*: The client has requested too many pieces of scrip (see below).
- *Bad_broker*: The server does not recognise the given broker.
- *Too_many_brokers*: The server accepts only one broker per request.

guarantee

A vendor may guarantee to provide a service at an offered price until a given time. The format of a guarantee is:

```
price "/" vendor-name "/" expiration-date "/" URL "<" vendor-signature ">"
```

where *protocol-name* is the full name of the payment protocol, *URL* is the identity of the requested item, and *vendor-signature* certifies the guarantee to the vendor for the requested URL.

offer-info

A vendor may offer to sell an item at a given price as described in an *offer-info* field. The format of an offer-info is:

```
price "|" broker ["|"broker]*
```

The offer includes a list of one or more brokers where vendor scrip in that currency may be purchased. An offer-info is only advisory unless accompanied by a guarantee.

price-info

The price-info describes the amount paid for a purchase. It consists of a *price*.

receipt-info

Millicent does not supply receipt information.

Field elements

This section describes elements that are common to more than one field.

broker/vendor

The format for a broker or vendor identifier is:

```
DNS-name | IP number ":" port-number
```

For example, identifiers might be: `mcbroker.pa.dec.com:2000` and `16.1.0.2:2005`

customer secret

To permit authenticated communications, a customer secret is associated with a customer identifier for a chosen vendor. The customer secret is used in the governing secret used to sign messages. The customer receives the customer secret when originally buying vendor scrip from a broker.

Brokers and vendors need to agree on a technique for recovering the customer secret from the customer id. The customer secret is constructed by using the master customer secret as the HMAC-MD5 key and hashing the customer's identifier. Both the customer identifier and customer secret are valid only for the relevant vendor. Note that the master customer secret is not revealed to the customer; it is used to construct customer secrets for many customers.

date

All dates are in Universal Time (a.k.a. Greenwich Mean Time) in reverse order, ie. starting with large units (years) and working down to small units (seconds and smaller). The format is:

```
YYYY[MM][DD][hh][mm][ss]
```

where *YYYY* is a four digit year, *MM* is a two digit month, *DD* is a two digit day of month, *hh* is a two digit hour in 24-hour notation, and *mm* and *ss* are two digit minutes and seconds respectively. There is no whitespace or punctuation, and all two digit numbers have a leading zero if the value is less than 10. A two digit number with value zero is represented by "`00`" unless it is the last field on the right of the string, in which case it is omitted. For example, the date "12:40:10 7 Dec 1994 GMT" is represented by: `19941207124010`.

This approach means that date comparisons reduce to string comparisons on most current systems.

governing secret

Authorised messages are signed by hashing a portion of the message and using the governing secret as the key. The governing secret for a transaction is constructed from the customer secrets of the pieces of scrip used in initiating the transaction.

If only a single piece of scrip is used, the governing secret is the associated customer secret.

If multiple pieces of scrip are used, the governing secret is the hash (with just MD5) of the concatenation of the customer secrets of the pieces of scrip, concatenated in the order determined by sorting the scrip-id's in ascending ascii order. The governing secret is used to sign a request, the corresponding reply, and to construct the one-time pad used in returning new-customer-secret.

hashing

All hashing uses 128 bit MD5 (RFC 1321) and HMAC-MD5 (internet draft) algorithms encoded in ascii base 16. That is, the 128 bits hash result is represented as 32 ASCII printable characters. The bits in the 128 bit digest are converted from most significant to least significant bit, four bits at a time to their ASCII presentation as follows. Each four bits is represented by its familiar hexadecimal notation from the characters `0123456789abcdef`. That is binary 0000 gets represented by the character '0', 0001, by '1', and so on up to the representation of 1111 as 'f'.

All keyed hashes calculations (hashes where there is an identifiable *secret* used as the *key*) use HMAC-MD5 to calculate the hash. These include the request and reply authorizations, customer secrets, and new customer secrets.

The governing secret calculation is made using a straight MD5 algorithm.

new-customer-secret

A server may need periodically to send a customer a new customer secret for a set of scrip. To reduce security costs, the new secret is encrypted using the governing secret as follows: for a governing secret *Gov* and a new secret *New*, the server generates a unique nonce *Nonce* which should include a human-readable timestamp. *New* and *Gov* are 128 bits long hash keys encoded in ascii base 16. The transmitted value is then:

```
Nonce "|" (New XOR Hash(Nonce^Gov)).
```

That is, the nonce followed by the new secret Xor'ed with the hash of the concatenation of the nonce and the governing secret; there is no whitespace in this field. The nonce contains only alphanumeric characters and Xor'ed secret is encoded in ascii base 16.

A server may wish to spontaneously return scrip in the reply to a non-Millicent request. In this case, the governing secret used to authenticate the reply is the string "00000000000000000000000000000000" (i.e. 32 zeroes, corresponding to the null secret).

price

Prices are encoded as character strings consisting of a number followed by a currency code.

These currency codes are the three letter ISO 4217 codes, Internet Assigned Number Authority (IANA) registered four letter or longer currency codes, or private use currency codes starting with "x-". (ISO 4217 code normally consist of the two letter country code followed by a letter mnemonic for the major unit of currency.) Currency codes are case insensitive. One and two letter codes appearing in the place of a country code are reserved for future use. A currency code can contain the alphanumeric characters, '-' and '_', and must start with a alphabetic character.

The number preceding the currency designation is the quantity of units of that currency. The quantity optionally may include a decimal point and decimal fraction. No leading zeroes are allowed unless the value is smaller than one currency unit, in which case there must be a single zero before the decimal point. No trailing zeroes are allowed; if the value is a round number of currency units, then the decimal point and fraction must be omitted. Values of any degree of accuracy may be specified, but implementations are only required to guarantee 9 significant digits. In addition, each currency will have a default accuracy which is normally 0.01 of the minor unit, if there is one, otherwise 0.01 of the major unit. So, US dollars are accurate to 0.01 of a cent, whereas Italian lire are accurate to 0.01 of a lira.

For example: `2.34gbp, 79ALL, 1.23456cad, 0.125usd` signifies 2 pounds and 34 pence sterling, seventy nine Albanian Leks, one dollar and 23 and 456 thousandths cents Canadian, and one eighth of a US dollar.

signature

All signatures are hashes of the signed data.

vendor name

a *vendor name* is a string of printable characters excluding those in "`/,`" and starting with an alphabetic character.

vendor signatures

a *vendor signature* is a string that certifies some item to the vendor. The certification technique is private to the vendor, the customer just passes back the string it receives with the item. As with hashes, vendor signatures are encoded in ASCII Base16.

Reserved URLs

In general, a Millicent "meta-transaction" will return one or more pieces of scrip to the customer. These are included in the HTML header of the reply, each piece enclosed within `"<Millicent-Scrip>"` and `"</Millicent-Scrip>"` tags.

The page at `"http://"`*`server`*`"/payment.000/Millicent/` contains a human-readable description of the versions of Millicent that the server supports. Otherwise, all Millicent transactions take place within a version-specific URL, for example: `"http://"`*`server`*`"/payment.000/Millicent-1.0/"`

A customer when purchasing broker or vendor scrip is sent a secret which the customer uses to sign purchase requests. This secret is included in the HTML header of the reply enclosed within `"<Millicent-Secret>"` and `"</Millicent-Secret>"` tags. Where multiple pieces of scrip are returned in a single reply, they all use the same secret. Messages with secrets must be communicated securely, rather than in the clear.

A customer contacts a vendor: to purchase items, convert vendor scrip, return vendor scrip, and refresh vendor scrip. A customer contacts a broker: to acquire broker scrip, to acquire vendor scrip, to return broker scrip, and to refresh broker scrip.

Within the payment URL space, a server is required to implement the following addresses:

```
"/payment.000/Millicent-1.0/convert" ["?" ["currency="<currency-code>]
["&pieces="<number-pieces>] ["&broker="<broker>] ]
```

This address allows customers to manipulate pieces of scrip. There are three aspects to a conversion:

between currencies

scrip may be converted from one currency to another, the target currency being defined in the *currency-code* field of the URL. If there is no currency field, then the server may return any currency, although it will normally try to match a currency in the scrip submitted with the request. If the server cannot provide the requested currency, it returns the `403 Forbidden` error with `Bad_currency` in a Payment-Error header.

changing the number of pieces

The *number-pieces* field determines how many nearly equal value pieces of scrip the server returns -- how equal the pieces are depends on the server. If not present, its value defaults to one, otherwise it must be a positive integer. If a customer requests more pieces than the vendor is prepared to supply, perhaps because the resulting pieces would be too small, the vendor returns the `403 Forbidden` error with `Too_many_pieces` in a Payment-Error header.

If *number-pieces* is `"0"`, then the customer is returning the enclosed scrip. Vendor scrip must be returned to the vendor from which it came and is returned as broker scrip.

changing the scrip broker

A customer may send scrip from multiple brokers in a request, so the server must chose one broker for the returned scrip. The client can specify this in the *broker* field or the server will pick a broker from one of the submitted pieces of scrip. If the server does not recognise the given broker, it returns the `403 Forbidden` error with `Bad_broker` in a Payment-Error header.

A server need not support requests with multiple brokers in which case a request with scrip from multiple sources will generate the `403 Forbidden` error with `Too_many_brokers` in a Payment-Error header.

A server may charge for a conversion, so the request must include a Payment-Bid header to be used for the conversion cost (even if the bid is zero) and the reply may include a Payment-Receipt header. Apart from any approved charges, all the scrip in a request will be returned or converted into *number-pieces* return scrips.

A broker is also required to support the following URLs

`/payment.000/Millicent-1.0/vendor`

This address returns a human-readable page that allows a customer to discover the vendors that this broker supports.

`"/payment.000/Millicent-1.0/vendor?vendor="<vendor>"&value="<value>["&pieces="<numbe`

This address allows customers to buy vendor scrip from a broker. *vendor* specifies the relevant vendor, and *value* specifies the amount to buy using the price format. The request must include a Payment-Bid header that gives the maximum amount the customer wants to pay for the requested vendor scrip, include any charges. The broker scrip to pay for the vendor scrip is sent in `Payment-Cash` headers. The *number-pieces* field is the same as in the the `convert` request. If the vendor is not supported, the broker will return the `404 Not Found` error.

The change (in broker scrip) and the purchased vendor scrip are returned in the reply's

`Payment-Cash` headers.

$Id: mcproto.htm,v 1.3 1995/11/06 20:50:40 sfreeman Exp sfreeman $

The Millicent Protocol for Inexpensive Electronic Commerce

Steve Glassman
Mark Manasse
Martín Abadi
Paul Gauthier
Patrick Sobalvarro

Proc 4th International
WWW Conf.
Dec 1995

Abstract:

Millicent is a lightweight and secure protocol for electronic commerce over the Internet. It is designed to support purchases costing less than a cent. It is based on decentralized validation of electronic cash at the vendor's server without any additional communication, expensive encryption, or off-line processing.

The key innovations of Millicent are its use of *brokers* and of *scrip*. Brokers take care of account management, billing, connection maintenance, and establishing accounts with vendors. Scrip is digital cash that is only valid for a specific vendor. The vendor locally validates the scrip to prevent customer fraud, such as double spending.

Keywords:

electronic commerce, electronic cash, scrip, broker, authentication

Contents

- Electronic Commerce Background
- Millicent
- Security and Trust
    - Trust model
    - Security
- Scrip
    - Scrip structure
    - Validation and expiration
    - Properties
- Millicent Protocols
    - Scrip in the clear
    - Private and secure
    - Secure without encryption
- Brokers
    - Scrip warehouse
    - Licensed scrip production
    - Multiple brokers
- Customer, Broker, and Vendor Interactions
- Status
- Future and Applications
- Conclusion

- Appendix
- References
- About the Authors

Electronic Commerce Background

There are a number of existing and proposed protocols for electronic commerce, such as those from DigiCash [2], Open Market [14], CyberCash [1], First Virtual [3], and NetBill [12]. They are all appropriate for medium to large transactions, $5 or $10 and up, because the costs per transaction are typically several cents plus a percentage. When these costs are applied to inexpensive transactions, 50 cents and less, the transaction costs become a significant or even dominant component of the total purchase price, thereby effectively creating a minimum price for goods and services purchased using one of these protocols.

Forcing online charges to be above some threshold reduces the options for service providers. Online services providing newspapers, magazines, reference works, and stock prices all have individual items that could be inexpensive if sold separately. The ability to purchase inexpensive individual items would make these services more attractive to casual users on the Internet. In addition, secure low-priced transactions support grass-roots electronic publishing. A user who is not likely to open a ten-dollar account with an unknown publisher may be willing to spend a few cents to buy an interesting-looking article.

In this section, we look at four existing options for Internet commerce: accounts, aggregation, credit cards, and digital cash, and discuss why they are not appropriate for inexpensive electronic commerce. In the next section, we describe our model for reducing costs and making lightweight electronic commerce feasible.

- Accounts

  The simplest model for electronic commerce is for customers to establish accounts with vendors. When a customer wants to perform a transaction with the vendor, the customer identifies himself (securely) and the vendor adds the cost of the transaction to the customer's account. Vendors maintain the account information and bill the customers periodically.

  With accounts, transaction costs and prices can be fairly low, but there is a fair amount of overhead. An account may need to be established ahead of time and maintained over an extended period. This makes sense only when assuming a relatively long-standing relationship between a customer and a vendor. There is often a minimum monthly charge associated with each account. The customer has separate accounts for each vendor, and the vendor needs to maintain accounts for every customer. All this overhead discourages casual users from making spur-of-the-moment purchases.

- Aggregation

  Aggregation amortizes billing charges over a sequence of less expensive transactions by accumulating transactions at the vendor until they exceed some threshold. Aggregation is another form of accounts and shares some of the problems of accounts. Although account setup is somewhat simplified, the vendor still has the problem of maintaining the accounts, accumulating enough transactions for a reasonable sized charge, and keeping transaction records for dispute resolution. Also, the customer must deal with separate charges from each vendor, minimum account charges, and the difficulty of contesting fraudulent charges.

- Credit cards

  Another simple model for electronic commerce is to use a credit card to pay for the purchase. Customers have credit cards; vendors register with credit card companies; customers give their credit card number to vendors; vendors contact their credit card companies for payment; the credit card companies handle the accounting and billing. There are established methods (like Netscape's SSL [13] based on RSA's public key encryption [16]) for ensuring secure transmission of the client's credit card number to the vendor.

  Unfortunately, credit card transactions are (relatively) expensive since every purchase involves communication to a centralized credit card transaction service. In addition, credit card companies offer various features like individual item accounting, insurance, and fraud protection that add to the cost and aren't needed when purchasing inexpensive items.

  Finally, customers may be unwilling to provide a credit card number to a vendor they don't know well. Although the credit card company insures the customer against any loss, there is still the inconvenience of clearing up any problems.

- Digital cash

  Digital cash is normally issued by a central trusted entity (like a bank). The integrity of digital cash is guaranteed by the digital signature of the issuer, so that counterfeiting digital cash is extremely hard. However, it is trivial to duplicate the bit pattern of the digital cash to produce and spend identical (and equally authentic) cash.

  In an online digital cash scheme, when a vendor receives digital cash, he must contact the issuer to see if it is valid and not already spent. This extra communication makes the central site a bottleneck and adds cost to the transaction.

  In an off-line scheme (like one proposed by DigiCash [2]), the vendor authenticates the digital cash during the transaction and then later transmits it to the issuer to check for double spending. This scheme adds computational costs to the vendor for authenticating the digital cash, and adds messages and encryption to the protocol for pinpointing the source of the double spending.

Millicent

Our goal for Millicent is to allow for transactions that are inexpensive yet secure. We achieve this by using accounts based on scrip and brokers to sell scrip.

A piece of scrip represents an account the customer has established with a vendor. At any given time, a vendor has outstanding scrip (open accounts) with the recently active customers. The balance of the account is kept as the value of the scrip. When the customer makes a purchase with scrip, the cost of the purchase is deducted from the scrip's value and new scrip (with the new value/account balance) is returned as change. When the customer has completed a series of transactions, he can "cash in" the remaining value of the scrip (close the account).

Brokers serve as accounting intermediaries between customers and vendors. Customers enter into long-term relationships with brokers, in much the same way as they would enter into an agreement with a bank, credit card company, or Internet service provider. Brokers buy and sell vendor scrip as a service to customers and vendors. Broker scrip serves as a common currency for customers to use when buying vendor scrip, and for vendors to give as a refund for unspent scrip.

Millicent reduces the overhead of accounts in a number of ways:

- Communication costs are reduced by verifying the scrip locally at the vendor's site; there are almost no Millicent-specific communication costs during a normal transaction. There is also no need for a centralized server or an expensive transaction-processing protocol.

  In a centralized scheme, the central site is a bottleneck; the provider must have sufficient computing power to handle the peak transaction rate. In Millicent, there is no central server; there can be many brokers, a broker is only involved in a fraction of the transactions between a customer and a vendor, and the transactions involving a broker are lightweight.

- Cryptographic costs are reduced to keep them in line with the scale of transactions; we don't need strong or expensive cryptographic schemes because the value of the scrip is relatively low. We need only make the cost of breaking the protocol greater than the value of the scrip itself.

- Accounting costs are reduced by using brokers to handle accounts and billing. The customer establishes an account with a broker; the broker establishes its own accounts with the vendors. Using brokers allows us to split a customer-vendor account into two accounts: one between the customer and broker, and another between the broker and the vendor. This reduces the total number of accounts. Instead of many separate accounts for every customer-vendor combination, each customer has only one account with a broker (or, at most, a couple of brokers); and each vendor has long-standing accounts with just a few brokers.

  In most account-based schemes, the vendor maintains the account balance. In Millicent, the customer maintains the account balance -- it is encoded in the scrip held by the customer. There is no risk for the vendor because a digital signature prevents the customer from modifying the scrip's value. Since the scrip contains the account balance and a proof of correctness for that value, the vendor does not need to look up the customer's balance, saving disk activity.

- The minimum monthly charges are not as much of a problem because they are amortized over more activity. The single customer-broker account supports transactions with all vendors and so it is likely to have enough activity to cover a minimum charge. By pre-paying the broker, even the monthly accumulation of charges can be avoided.

Millicent is best suited for a series of inexpensive, casual transactions. We will rely on other protocols for initial account establishment between brokers and customers, and brokers and vendors. Other higher-value protocols are also used for the funds transfers that occur when accounts are periodically settled.

Security and Trust

The security model for Millicent is based on the assumption that scrip is used for small amounts. People and businesses treat coins differently than they treat bills, and treat small bills differently than large bills. In Millicent, we imagine people treating scrip as they would treat change in their pocket.

Since people don't need a receipt when buying candy from a vending machine, they don't need a receipt when buying an item using scrip. If they don't get what they paid for, they complain and get a refund. If they lose a coin every now and then, they aren't too upset.

We expect users to have a few dollars of scrip at a time. We don't expect them to have hundreds, or even tens, of dollars of scrip. As a result, scrip is not worth stealing unless you can steal lots of it; and if you steal lots, you will get caught.

Trust model

Millicent assumes asymmetric trust relationships among the three entities - customers, brokers, and vendors. Brokers are assumed to be the most trustworthy, then vendors, and, finally, customers. The only time customers need to be trusted is when they complain about service problems.

We believe that brokers will tend to be large, well-known, and reputable financial institutions (like Visa, MasterCard, and banks) or major Internet or online service providers (like CompuServe, NETCOM, or AOL). We expect there to be many vendors covering a full spectrum of size and trustworthiness, as in the real world. Finally, there will be large numbers of customers who are as trustworthy as people are in general.

Three factors make broker fraud unprofitable. First, customer and vendor software can independently check the scrip and maintain account balances, so any fraud by the broker can be detected. Second, customers do not hold much scrip at any one time, so a broker would have to commit *many* fraudulent transactions to make much of a gain and this makes them likelier to be caught. Finally, the reputation of a broker is important for attracting customers and a broker would quickly lose its reputation if customers have troubles with the broker. The repeat business of active customers is more valuable to a broker than the scrip that it could steal.

Vendor fraud consists of not providing goods for valid scrip. If this happens, customers will complain to their broker, and brokers will drop vendors who cause too many complaints. This acts as an effective policing mechanism, because vendors need a broker to easily conduct business in Millicent.

As a result, the Millicent protocol is skewed to prevent customer fraud (forgery and double spending) while providing indirect detection of broker and vendor fraud.

Security

The security of Millicent transactions comes from several aspects.

- All transactions are protected

  Every Millicent transaction requires that the customer knows the secret associated with the scrip. The protocol never sends the secret in the clear, so there is no risk due to eavesdropping. No piece of scrip can be reused, so a replay attack will fail. Each request is signed with the secret, so there is no way to intercept scrip and use the scrip to make a different request.

- Inexpensive transactions limit the value of fraud

  Inexpensive transactions can rely on inexpensive security: it's not worth using expensive computer resources to steal inexpensive scrip. In addition, it would take many illegal uses of scrip to acquire much money, and that raises the probability of getting caught.

- Fraud is detectable and eventually traceable

  Fraud is detected when the customer doesn't obtain the desired goods from the vendor, or when the balance returned to the customer doesn't match the balance due. If the customer is cheating, then the vendor's only loss is the cost of detecting the bad scrip and denying service. If the vendor is cheating, the customer will report a problem to the broker. When a broker notices a pattern of complaints from many customers against a vendor, it can pinpoint the fraud and cut off all dealings with the vendor. If a broker is cheating, the vendor will notice bad scrip coming from many customers, all originating from a single broker. The vendor can then publicize its complaint in an appropriate venue.

Scrip

The main properties of scrip are:

- It has value at a specific vendor.
- It can be spent only once.
- It is tamper resistant and hard to counterfeit.
- It can be spent only by its rightful owner.
- It can be efficiently produced and validated.

The next sections give more detail about scrip and its use, but the basic techniques to achieve these properties are outlined here:

- The text of the scrip gives its value and identifies the vendor.
- The scrip has a serial number to prevent double spending.
- There is a digital signature to prevent tampering and counterfeiting.
- The customer signs each use of scrip with a secret that is associated with the scrip.
- The signatures can be efficiently created and checked using a fast one-way hash function (like MD5 [15] or SHA [11]).

Scrip structure

There are three secrets involved in producing, validating, and spending scrip. The customer is sent one secret, the `customer_secret`, to prove ownership of the scrip. The vendor uses one secret, the `master_customer_secret`, to derive the `customer_secret` from customer information in the scrip. The third secret, the `master_scrip_secret`, is used by the vendor to prevent tampering and counterfeiting.

The secrets are all used in a way that shows knowledge of the secret without revealing the secret. To attest to a message, the secret is appended to the message, and the result is hashed to produce a signature. The message (without the secret) and the signature prove - due to the one-way nature of the hash function - knowledge of the secret, because the correct signature can only be derived if you know the secret.

Scrip has the following fields (Figure 1):

- `Vendor` identifies the vendor for the scrip.
- `Value` gives the value of the scrip.
- `ID#` is the unique identifier of the scrip. Some portion of it is used to select the `master_scrip_secret` used for the certificate.
- `Cust_ID#` is used to produce the customer secret. A portion of `Cust_ID#` is used to select the `master_customer_secret` which is also used in producing the customer secret.
- `Expires` is the expiration time for the scrip.
- `Props` are extra data describing customer properties (age, state of residence, etc.) to the vendor.
- `Certificate` is the signature of the scrip.

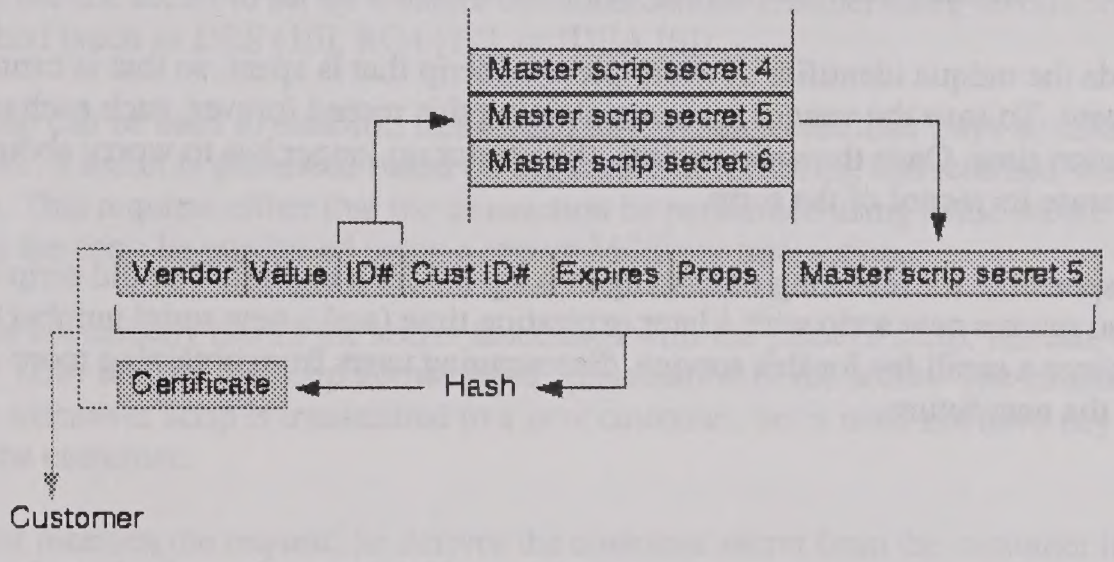


Figure 1: The certificate of a piece of scrip is generated by hashing the body of the scrip with a secret. The secret is selected using a portion of the scrip's ID#.

Validation and expiration

Scrip is validated in two steps. First (Figure 2), the certificate is recomputed and checked against the certificate sent with the scrip. If the scrip has been tampered with, then the two certificates will not match. Second, there is a unique identifier (ID#) included in the scrip body and the vendor can check for double spending by seeing if it has recorded that identifier as already spent. Generating and validating scrip each require a little text manipulation and one hash operation. Unless the secret is known, scrip cannot be counterfeited or altered.

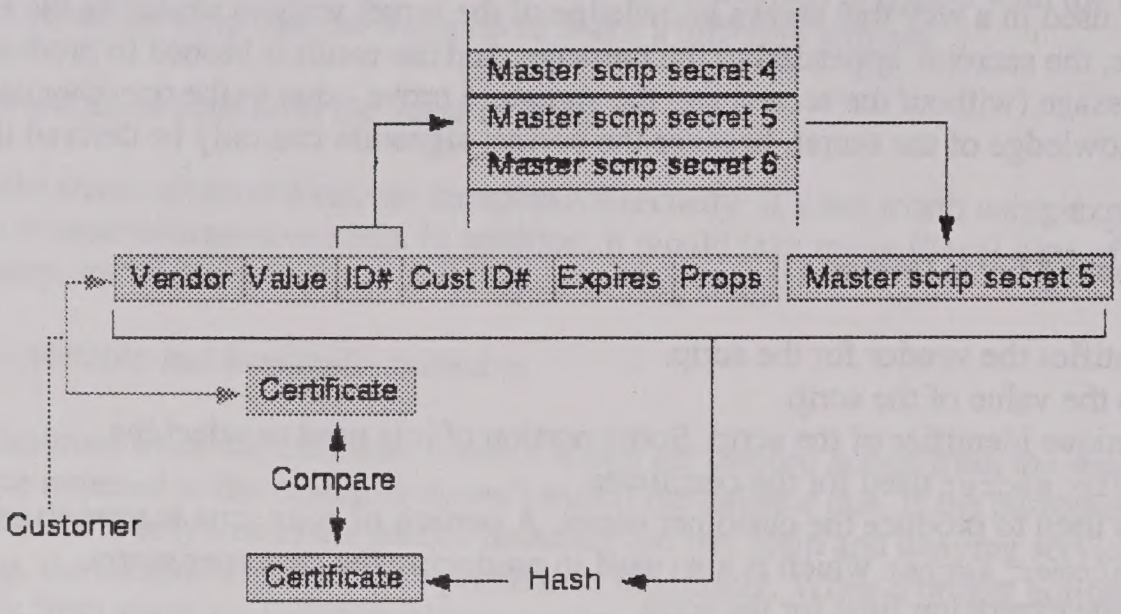


Figure 2: The received scrip is validated by regenerating the certificate and comparing it to the transmitted one. If they are identical, the scrip is valid.

The vendor records the unique identifier of every piece of scrip that is spent, so that is cannot be fraudulently re-spent. To save the vendor from maintaining this record forever, each each piece of scrip is given an expiration time. Once the scrip expires, the vendor no longer has to worry about it being re-spent and can erase its record of the scrip.

Customers are responsible for renewing or cashing in scrip before it expires. The old scrip is submitted to the vendor, who returns new scrip with a later expiration time (and a new serial number). Vendors may choose to charge a small fee for this service, discouraging users from obtaining more scrip than they will need in the near future.

Properties

Scrip also has fields for storing properties, which are inserted by the vendor or broker when the scrip is produced. The exact property fields and their values will depend on an agreement between the brokers and vendors. The brokers will get the information from customers when they create their account and enforce some set of rules when selling vendor scrip. Vendors, of course, are free to include whatever properties they desire in scrip they produce themselves.

Information such as the state of residence, or age of the consumer assists the vendor in making sales decisions. Adult material could only be bought if the scrip shows the customer is old enough. State sales tax charges can depend on a property included in the scrip.

Millicent Protocols

Scrip is the basis of a family of Millicent protocols. We will describe three of them and compare their simplicity, secrecy, and security. (A detailed description of the protocols is in the appendix.)

The first, "scrip in the clear", is the simplest and most efficient protocol. It is the basis for the other two protocols, but it may not be useful in practice because it is too insecure. The second, "private and secure", is secure and offers good privacy, but it is more expensive. The third, "secure without encryption", is also secure, but trades privacy for greater efficiency.

Scrip in the clear

In the simplest possible Millicent protocol, the customer just sends an unspent piece of scrip in the clear (i.e., not encrypted or protected in any way) along with each request to the vendor. The vendor returns the desired result along with a new piece of scrip (also in the clear) as change.

This protocol offers almost no security: an eavesdropping third party can intercept the scrip being returned as change and use it himself. When the rightful owner later attempted to spend the scrip, the vendor would have a record of it being previously spent, and would refuse the request.

Private and secure

To add security and privacy to the Millicent protocol, we establish a shared secret between the two parties and then use the secret to set up a secure communications channel using an efficient, symmetric encryption method (such as DES [10], RC4 [17], or IDEA [6]).

In Millicent, scrip can be used to establish this shared key. When a customer buys an initial piece of scrip for a vendor, a secret is generated based on the customer identifier, and returned securely with the scrip (Figure 3). This requires either that the transaction be performed using some secure non-Millicent protocol, or that the scrip be purchased using a secure Millicent transaction.

The vendor does not directly record the secret associated with the piece of scrip. Instead, the customer identifier (Cust_ID#) field of the scrip allows rapid recalculation of the secret. The customer identifier must be unique whenever scrip is transmitted to a new customer, but it need not have any connection to the identity of the customer.

When the vendor receives the request, he derives the customer secret from the customer identifier in the scrip, derives the message key from the customer secret, and uses the message key to decrypt the request. The change scrip can be returned in the clear, while the response and any new secrets are returned to the customer encrypted by the message key.

In this protocol the request and the response are kept totally private; unless an eavesdropper knows the customer secret, he can't decrypt the messages. In addition, an eavesdropper can't steal the scrip because it can't be spent without knowing the customer secret.

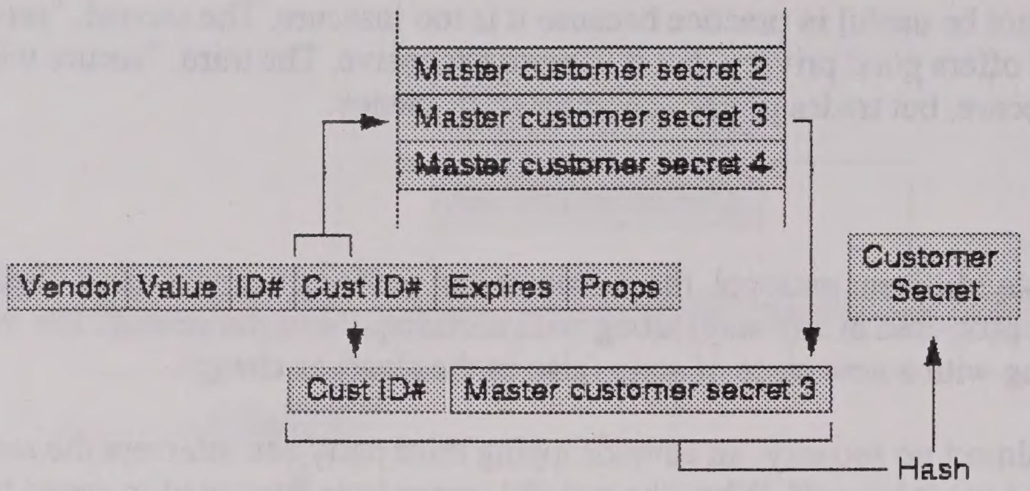

Master customer secret 2
Master customer secret 3
Master customer secret 4
Vendor Value ID# Cust ID# Expires Props
Customer Secret
Cust ID#
Master customer secret 3
Hash

Figure 3: The customer secret is generated by hashing the customer identifier with a secret. The secret is selected using a portion of the customer identifier.

Secure without encryption

The previous section describes how the secret shared by the customer and vendor can be exploited to achieve security and privacy. But a full-blown encrypted channel may be overkill for some Millicent applications. In this, our third variant of the protocol, we give up the privacy of the request and response to eliminate the use of encryption.

As in the previous protocol, the customer securely gets an initial piece of scrip and customer secret. To make a purchase, the customer sends the request, scrip, and a "signature" of the request to the vendor. The signature is produced in the same way that the certificate of the scrip is produced. The scrip and request are concatenated with the customer secret. The customer runs an efficient cryptographic one-way hash function over this string and sends the resulting hash as the signature.

When the vendor receives the request, he derives the customer secret from the scrip and regenerates the signature for the request. If the scrip or request have been tampered with in any way, the signature will not match. (Figure 4).

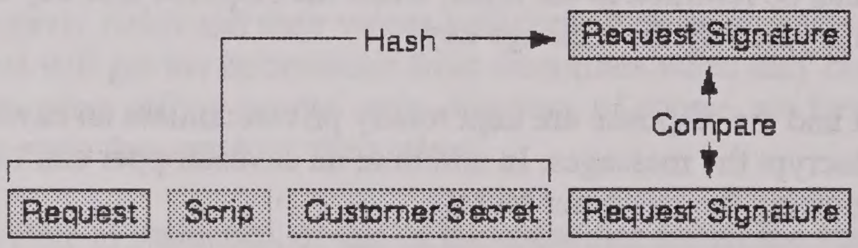

Hash
Request Signature
Compare
Request
Scrip
Customer Secret
Request Signature

Figure 4: The request is validated by re-generating the request signature and comparing to the transmitted signature. If they match, the request is valid.

The vendor now handles the request and returns a fresh piece of scrip as change. The change scrip shares the same customer identifier as the scrip submitted with the request, so that the original customer secret can be used to spend the change. There is no need to encrypt any of the response; an eavesdropper can't steal the scrip because the signature of the request can't be made without knowing the customer secret. The vendor may sign the response with the customer secret in order to prove authenticity to the customer.

Thus, with only a few hashes, Millicent provides a lightweight and secure protocol.

Brokers

Brokers maintain the accounts of customers and vendors, and they handle all real-money transactions. The customer establishes an account with a broker by using some other method (like a credit card, or a higher-security electronic commerce system) to buy some broker scrip. The customer then uses the broker scrip to buy vendor scrip.

The vendor and the broker have a long-term business relationship. The broker sells vendor scrip to customers and pays the vendor. There can be different business models for the way the broker gets vendor scrip, for example, pay in advance, consignment sale, or licensed production. In all models, the broker can make a profit selling scrip because he pays the vendor (at a discount) for scrip in bulk and sells individual pieces to customers.

When a customer wants to make a purchase, the customer contacts the broker to obtain the necessary vendor scrip. The customer uses his broker scrip to pay for the vendor scrip using the Millicent protocol. The broker returns the new vendor scrip along with change in broker scrip.

We will examine three ways in which the broker gets the vendor scrip. The "scrip warehouse" model assumes a casual relationship between the broker and vendor. The "licensed scrip producer" model assumes a substantial and long-lasting relationship between the broker and vendor. The "multiple broker" model assumes a relationship between brokers, but requires no relationship between the vendor and broker.

Scrip warehouse

When the broker is acting as a scrip warehouse, the broker buys multiple pieces of scrip from a vendor. The broker stores the scrip and sells the pieces one at a time to customers (Figure 5b-5d).

This model assumes no special relationship between the vendor and broker. It works best when the broker's customers have a light to moderate demand for that vendor's scrip. The broker uses the Millicent protocol to buy the scrip from the vendor in the same way a customer would. Selling scrip in large blocks is more efficient for the vendor since the communication and financial transaction costs are amortized over all the pieces of scrip. We presume that the vendor offers some sort of volume discount to encourage brokers to buy large blocks of scrip. The broker makes a profit when it resells the scrip to customers at full price. The vendor depends on the broker to ensure any customer properties encoded in the scrip.

Licensed scrip production

If a broker's customers buy a lot of scrip for a specific vendor, it may be desirable for a vendor to "license" the broker to produce vendor scrip. This means that the broker generates scrip that the vendor can validate and accept. The vendor sells the the broker the right to generate scrip using a given `master_scrip_secret`, series of scrip ID#'s, `master_customer_secret`, and series of customer identifiers. The vendor can validate the licensed scrip because the `master_scrip_secret` is known from the series of the scrip ID# and the `master_customer_secret` is known from the series of the customer identifier.

Brokers produce the scrip and collect money from customers; vendors record the total value of scrip originating from a particular broker. When all the scrip produced under a particular contract has expired, brokers and vendors can settle up. The broker presumably takes some commission for producing the scrip.

A license covers a specific series (unique range of identifiers - ID#'s) of scrip for a given period of time, and the secrets shared between the broker and vendor only apply to that series. A vendor can issue licenses to different brokers by giving out different series and secrets to each one. Of course, a vendor can produce its own scrip using its own private series and secrets.

Licensing scrip production is more efficient for the vendor and broker than the scrip warehouse model. There is less communication because the license is smaller to transmit than a few pieces of scrip. The vendor does less computation since it does not have to generate the scrip itself. The broker does not have to store large blocks of scrip, since it can generate the scrip on demand. Additionally, it allows the broker to encode specific user properties into each piece of scrip it generates.

Multiple brokers

In an environment where there are multiple brokers, a customer of one broker may want to make a purchase from a vendor associated with another broker. If the vendor only wants to have an account with its own broker (perhaps to simplify accounting), the customer will have to go through the vendor's broker to buy vendor scrip.

The entire transaction will go like this:

- The customer asks his broker for vendor scrip.
- The customer's broker tries to set up account with the vendor.
- The vendor tells the customer's broker his broker's name.
- The customer's broker buys broker scrip from the vendor's broker.
- The customer's broker returns the vendor's broker's scrip to the customer.
- The customer buys vendor scrip from the vendor's broker.
- The customer uses the vendor scrip at the vendor.

The idea of licensed scrip production can be extended so that brokers can generate broker scrip for other brokers.

Customer, Broker, and Vendor Interactions

The following diagrams (Figures 5a-e) present the steps for a complete Millicent session (including the broker buying scrip from the vendor). The initial step (Figure 5a) happens only once per session. The second step (Figure 5b) happens each time the customer has no stored scrip for a vendor. Step three (Figure 5c) happens only if the broker must contact the vendor to buy the scrip. It is not needed for licensed scrip production. The fourth step (Figure 5d) shows the broker returning the vendor scrip to the customer. The fifth step (Figure 5e) shows the customer using the scrip to make a purchase from the vendor.

The last step (Figure 5f) shows a typical Millicent transaction. The customer already has vendor scrip and uses it to make a purchase. There are no extra messages or interactions with the broker.

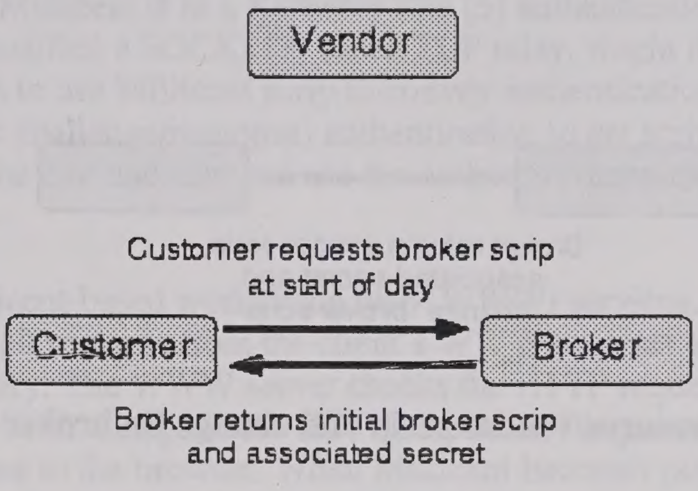

Vendor
Customer requests broker scrip at start of day
Customer
Broker
Broker returns initial broker scrip and associated secret

Figure 5a: The client makes a secure connection to the broker to get some broker scrip.

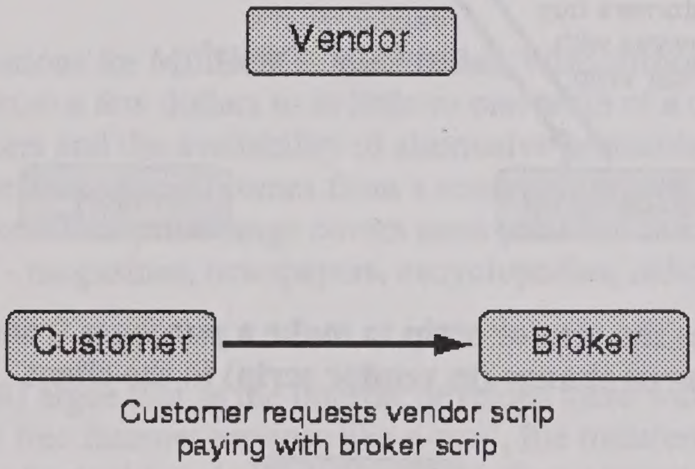

Vendor
Customer
Broker
Customer requests vendor scrip paying with broker scrip

Figure 5b: If the client doesn't already have scrip for a particular vendor, he contacts the broker to buy some using his broker scrip.

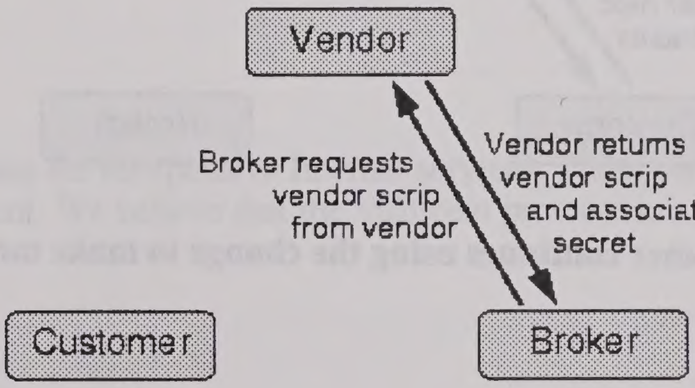

Vendor
Broker requests vendor scrip from vendor
Vendor returns vendor scrip and associated secret
Customer
Broker

Figure 5c: If the broker doesn't already have scrip for that vendor, he buys some from the vendor.

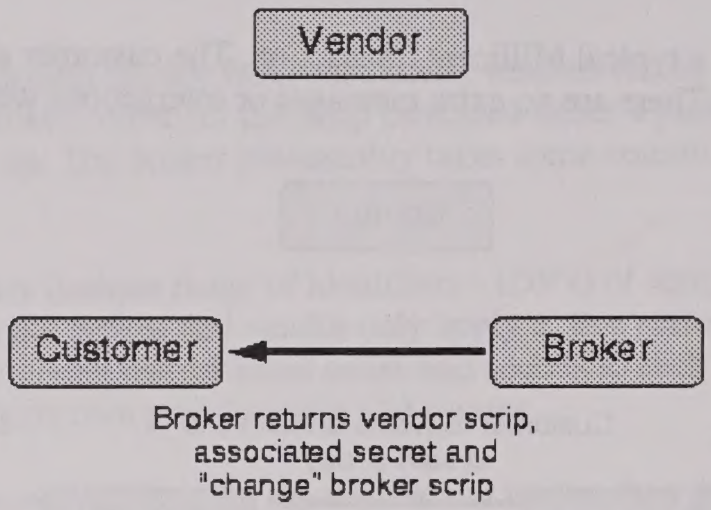

Vendor
Customer
Broker
Broker returns vendor scrip, associated secret and "change" broker scrip

Figure 5d: The broker returns vendor scrip and change (in broker scrip) to the client.

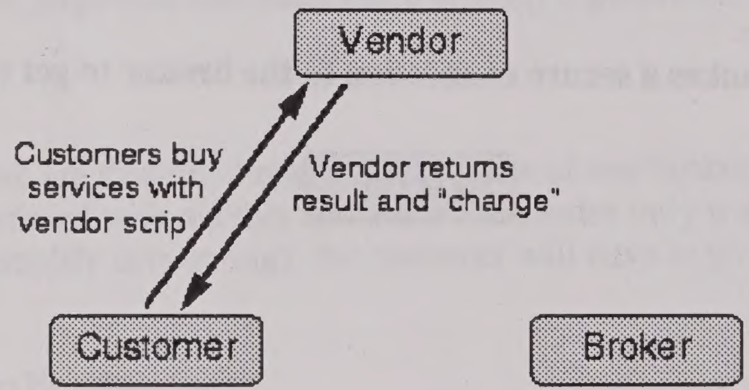

Vendor
Customers buy services with vendor scrip
Vendor returns result and "change"
Customer
Broker

Figure 5e: The customer uses the vendor scrip to make a purchase from the vendor. The vendor returns change (in vendor scrip) to the client.

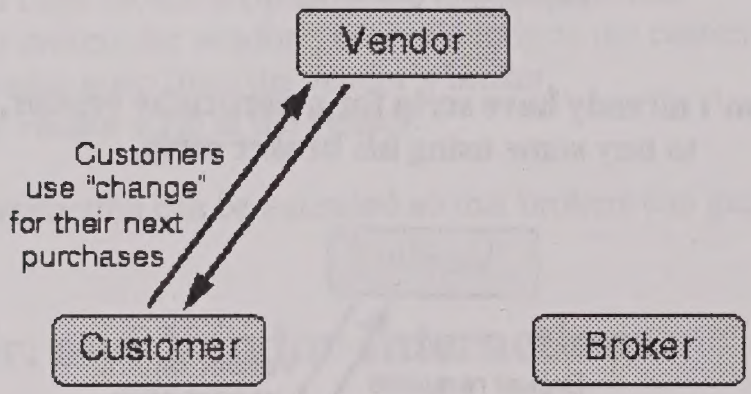

Vendor
Customers use "change" for their next purchases
Customer
Broker

Figure 5f: The customer continues using the change to make more purchases.

Status

We have produced an initial implementation of Millicent [9] consisting of a set of libraries, and a vendor and broker written using the libraries for Millicent transactions across a network using TCP/IP. Our measurements show that the Millicent protocol is efficient enough for sub-cent purchases. Our untuned vendor implementation can validate about 1000 Millicent requests per second (on a Digital AlphaStation 400 4/233) and, of that, most of the time goes into the TCP connection handling.

Using zero-cost transactions, Millicent scrip can be used as a distributed capability. Using this aspect of scrip, our first application of Millicent is in a Kerberos-like [5] authentication suite for our network firewall services. We have modified a SOCKs [7] based TCP relay, rlogin daemon, FTP daemon, and rlogin, telnet, and FTP clients to use Millicent scrip to convey authentication information. A user does one cryptokey (cryptographic challenge/response) authentication to get scrip from an authentication broker. Then, for the rest of the day, the user can use the authentication scrip to buy scrip for particular firewall services.

We are also working on Millicent-based world wide web (WWW) services. We have developed a local *pseudo-proxy* that intercepts all requests from the client's WWW browser and modifies the HTTP header to add scrip as necessary. The WWW server checks the HTTP request for sufficient scrip to buy the page and returns the page with change in the HTTP response. The pseudo-proxy extracts the change before forwarding the response to the browser. When Millicent becomes popular, the functionality of the pseudo-proxy can be integrated in with the browser.

Future and Applications

The range of potential applications for Millicent is quite broad. With current technology, Millicent is appropriate for transactions from a few dollars to as little as one-tenth of a cent. The upper bound comes from the trust model for brokers and the availability of alternative protocols appropriate for transactions above a few dollars, while the lower bound comes from a conservative estimate based on the computational costs of a broker. This price range covers most print and information services that will be available in an online format - magazines, newspapers, encyclopedias, indices, newsletters, and databases.

MacKie-Mason and Varian [8] argue that as the Internet develops there will be increasing pressure for usage-based charges. Current free Internet services like e-mail, file transfers, the Internet telephone, and tele-conferencing will have to be paid for. At the lowest level, they estimate that the cost of transmitting one packet on the Internet backbone is one six-hundredth of a cent. We don't believe that Millicent is quite efficient enough for such packet-level charges; for these there are proposals like the non-cryptographic *Digital Silk Road* [4]. We do believe that Millicent can be used for per-connection charges for these services.

Conclusion

We see growing opportunities for inexpensive Internet services. These services need an appropriate electronic commerce protocol. We believe that the Millicent protocol is a good candidate to be that protocol.

Appendix

The following is a more precise description of the Millicent protocol for customer and vendor. The interaction between customer and broker relies on the same protocol, as explained in the main body of the paper.

We use the following notations:

`X, Y, Z`
: represents the string encoding the tuple `X`, `Y`, `Z`.

`H(X)`
: is the result of hashing `X` with a cryptographic hash function, such as MD5.

`{X}Y`
: is the result of encrypting `X` with a cryptographic function, such as DES, under the key `Y`.

`A->B: X`
: means that `A` sends `X` to `B`.

Before giving the protocol, we describe its fields. First, we list some of the ingredients of scrip; scrip includes the name of the vendor, some properties of the customer, a value, and an expiration time:

`vendor_id:`
: A unique identifier (or name) for the vendor.

`props:`
: Any data describing customer properties (possibly including a name).

`value:`
: The value of the scrip.

`exp:`
: The expiration time for the scrip.

The customer and the vendor generate a request and a reply. Both of these are arbitrary strings.

`request:`
: The request from the customer.

`reply:`
: The reply from the vendor. We assume that, by its format, `reply` is distinguishable from `request`.

In addition, the protocol relies on various secrets, and on corresponding identifiers for those secrets:

`master_scrip_secret:`
: A secret used for certifying scrip. The `master_scrip_secret` is known only to the vendor (when the vendor produces his own scrip) or only to the vendor and to the broker (when the broker may produce scrip).

`id_series#:`
: An identifier for `master_scrip_secret`. The vendor can map `id_series#` to `master_scrip_secret`.

`id_sequence#:`
: A unique identifier, such as a sequence number. The vendor accepts `id_sequence#` for at most one transaction in conjunction with `id_series#`.

`id# = id_series#, id_sequence#`

`master_customer_secret:`
: A secret used for producing customer secrets. The `master_customer_secret` is known only to the vendor and to the broker.

`cust_id_series#:`
: An identifier for `master_customer_secret`. The vendor can map `cust_id_series#` to

`master_customer_secret.`

`cust_id_sequence#:`

A unique identifier, such as a sequence number. Together with `cust_id_series#`, it identifies the customer.

```
cust_id# = cust_id_series#, cust_id_sequence#
customer_secret = H(cust_id#, master_customer_secret)
```

A secret that the vendor or the broker sends to the customer. The customer gets this quantity along with `cust_id#` (but not `master_customer_secret`); the customer can map `vendor_id` and `cust_id#` to `customer_secret`. Both the vendor and the broker can generate `customer_secret` from `cust_id#` and `master_customer_secret`. No one else knows `customer_secret`.

Scrip is generated by combining all of the fields listed above, as follows:

```
id_material = vendor_id, id#, cust_id#
cert_material = props, value, exp
scrip_body = id_material, cert_material
cert = H(scrip_body, master_scrip_secret)
```

This is a certificate that proves the authenticity of `scrip_body` for the customer associated with `cust_id#`.

```
scrip = scrip_body, cert
```

The vendor or the broker gives `scrip` to the customer; the customer presents it to the vendor along with a request.

For change returned from a transaction, the vendor issues new scrip, with a new certificate. The new quantities may differ from the previous ones in all their components except for `cust_id#` and `vendor_id`. If the broker initially knows `master_scrip_secret` and this quantity remains the same, then the broker is in principle capable of producing change instead of the vendor. This may not be desirable, since it implies unnecessary trust from the vendor to the broker; hence, when the vendor makes change, it is sensible for the vendor to use a new `master_scrip_secret` not known to the broker. In any case, the vendor should pick a new value for `id#` as a protection against replays. We write `scrip'` for the new scrip.

- In the clear (insecure)

  ```
  customer -> vendor: scrip, request
  vendor -> customer: scrip', reply
  ```

- Authentic and private

  ```
  customer -> vendor: vendor_id, cust_id#, {scrip, request}customer_secret
  vendor -> customer: vendor_id, cust_id#, {scrip', cert, reply}customer_secret
  ```

  Most of the communication is under `customer_secret` for authenticity and privacy. It is possible to encrypt less, with a gain in efficiency. For example, some parts of `scrip'` are not sensitive and could be sent in the clear.

  The response includes `cert` in order to allow the customer to check that the response received is in fact a response to the request.

  Both messages include `vendor_id` and `cust_id#` in the clear in order to allow the recipient to generate `customer_secret`.

- Authentic but not private

```
customer->vendor: scrip, request, H(scrip, request, customer_secret)
vendor->customer: scrip', reply, H(scrip', cert, reply, customer_secret)
```

All messages are sent in the clear, but they are protected by the signatures (based on `customer_secret`).

No encryption is used and only five hashes are necessary at the server to handle a request. The hashes are: (1) for checking the old scrip, (2) for regenerating `customer_secret`, (3) for checking the customer's signature, (4) for generating the new scrip, and (5) for signing the response.

The response includes `cert` in order to allow the customer to check that the response received is in fact a response to the request.

Both messages include `vendor_id` and `cust_id#` in the clear (in `scrip` and `scrip'`) so the recipient can generate `customer_secret`.

References

1 CyberCash Inc., URL: http://www.cybercash.com/

2 DigiCash Inc., http://www.digicash.com/

3 First Virtual Holdings Inc., URL: http://www.fv.com/

4 Norman Hardy and Eric Dean Tribble, *The Digital Silk Road* URL: ftp://ftp.netcom.com/pub/jo/joule/DSR/DSR1.txt.gz or http://web.gmu.edu:80/bcox/Bionomics/Extropians/HardyTribbleSilkRoad.html

5 J. Kohl, C. Neuman, *The Kerberos Network Authentication Service (V5), IETF RFC 1510,* URL: gopher://ds2.internic.net/00/rfc/rfc1510.txt

6 Xuejia Lai, *On the Design and Security of Block Ciphers*, Institute for Signal and Information Processing, ETH-Zentrum, Zurich, Switzerland, 1992.

7 M. Leech, M. Ganis, Y. Lee, et. al., *SOCKS Protocol Version 5*, draft-ietf-aft-socks-protocol-v5-04.txt, URL: http://src.doc.ic.ac.uk/computing/internet/internet-drafts/draft-ietf-aft-socks-protocol-v5-04.txt.Z

8 Jeffrey K. MacKie-Mason, Hal R. Varian. *Some FAQs about usage based pricing*. University of Michigan, September 1994, URL: ftp://gopher.econ.lsa.umich.edu/pub/Papers/useFAQs.html

9 Mark S. Manasse, *A Method for Low Priced Electronic Commerce*, Patent pending, URL: http://www.research.digital.com/SRC/millicent

10 National Institute for Standards and Technology (NIST), *Data Encryption Standard (DES), Federal Information Processing Standards Publication 46-2*, December 1993, URL: http://www.ncsl.nist.gov/fips/fips46-2.txt

11 National Institute for Standards and Technology (NIST), *Secure Hash Standard, FIPS PUB 180-1: Secure Hash Standard*, April 1995, URL: http://csrc.ncsl.nist.gov/fips/fip180-1.txt

12 NetBill, Carnegie Mellon University, URL: http://www.ini.cmu.edu/netbill/

13 Netscape Inc., *SSL Protocol*, URL: http://home.netscape.com/newsref/std/SSL.html

14 Open Market Inc., URL: http://www.openmarket.com/

15 R. Rivest, *The MD5 Message-Digest Algorithm, IETF RFC 1321*, URL: gopher://ds2.internic.net/00/rfc/rfc1321.txt

16 RSA Inc., *PKCS#1: RSA Encryption Standard*, URL: http://www.rsa.com/pub/pkcs/ps/pkcs-1.ps

17 RSA Inc., *RSA's Frequently Asked Questions About Today's Cryptography*, URL: http://www.rsa.com/rsalabs/faq/faq_misc.html#misc.6

About the Authors

Martín Abadi [http://www.research.digital.com/people/Martin_Abadi/bio.html]
Systems Research Center, Digital Equipment Corporation
ma@pa.dec.com

Paul Gauthier [http://www.cs.berkeley.edu/~gauthier/]
University of California Berkeley
gauthier@cs.berkeley.edu
Paul Gauthier is a Ph.D. student at the University of Califoria, Berkeley. He was a summer intern at SRC in Summer, 1995.

Steve Glassman [http://www.research.digital.com/people/Steve_Glassman/bio.html]
Systems Research Center, Digital Equipment Corporation
steveg@pa.dec.com

Mark S. Manasse [http://www.research.digital.com/people/Mark_Manasse/bio.html]
Systems Research Center, Digital Equipment Corporation
msm@pa.dec.com

Patrick Sobalvarro [http://www.psg.lcs.mit.edu:80/~pgs/]
Massachusetts Institute of Technology
pgs@lcs.mit.edu
Patrick Sobalvarro is a Ph.D. student at the Massachusetts Institute of Technology. He was a summer

intern at SRC in Summer, 1995.

The Millicent protocols for electronic commerce

Mark S. Manasse
Systems Research Center
Digital Equipment Corporation
Palo Alto, California
msm@pa.dec.com
http://www.research.digital.com/SRC/people/Mark_Manasse/bio.html

Original Paper
July 1995

Table of Contents

- Introduction
- Millicent overview
- Millicent scrip
- Money
- Rekeying, refreshing, and redemption
- Problems
- Performance
- Extensions
- Conclusions
- Acknowledgments

Introduction

Many protocols have been proposed in the last year which address the problem of securely transferring money over a public network. Most of these schemes have similar goals: to enable transactions with properties similar to those achievable today using credit cards, to provide consumers and vendors with guarantees similar to those afforded by credit cards, or to translate existing payment mechanisms into electronic equivalents. The schemes vary widely in details; some schemes create electronic cash, some schemes work hard to provide anonymity for purchasers, some schemes protect consumers from exposing their underlying accounts to merchants, and some schemes ignore privacy issues altogether.

This is all interesting and necessary, but it fails to address the problems of a market segment that I expect to grow rapidly, once it becomes at all possible: extremely low-priced walk-up transactions. The proposed payment schemes all come with fee schedules that limit transactions to be fairly valuable. In practical terms, for today's systems, and today's mechanisms for transactions, the fees come to a minimum of approximately twenty-five cents for credit-card-like transactions, and at least a penny for services that provide a level of aggregation before charging a credit card. Over time these fees are almost certain to drop, but the current performance of disks, processors, and networks suggest that these fees aren't going to come down due to economic pressures alone: a system that can only handle a dozen transactions per second per computer needs to charge fees in this range in order to be profitable-and that's the kind of transaction rate implied by systems that require non-repudiatable digital signatures. Increased performance over time will increase the transaction rate; I contend that it's interesting to look at the results of applying that to decreasing the price of transactions.

With fees in their current range, the minimum transaction is likely to remain between five cents and one dollar. For tangible goods, this is unlikely to be a problem; even penny candy costs a few cents these days! For information goods, however, the costs for manufacturing and delivery are quite small. If the overhead for billing could be reduced to the same level, what changes might take place in information delivery? Does it matter if the smallest transaction is five cents or a mil?

I contend that it does matter. At five cents, one can decompose a daily newspaper into a few sections (sports, business, world news, local news) without significantly increasing the daily outlay for information. But this level of decomposition doesn't change the nature of information delivery. At a mil, we can decompose the newspaper into individual articles, comic strips, and horoscopes.

And this information marketplace isn't limited to newspapers: individual stock quotes, background stories culled from the morgue, technical papers, historical commercial information, index retrieval, almost anything where the information production can either be automated, or where the market is large enough. An anarchic information provision world, where payment can be rendered not only for the information, but for the filtering to provide just the information that's fit to download.

A decade hence, assuming that computers (and their components) continue on the price and performance curves of the last two decades, the minimum transaction grain will be an order of magnitude smaller than it is today. The smallest transactions will be in the sub-penny range. In a decade, we will see pricing by the web page. We can unbundle the newspaper and the magazine, derive revenue from the newspaper morgue, and sell information without restricting consumers to a subscription to a small number of sources. Citation and reference by hyperlink can replace quotation. "Fair use" photocopying can be replaced by links, and the information gets to the students that care at lower total cost, and with revenue returned to the author rather than the paper and toner companies. Editors, authors, and publishers become more independent, as an editor becomes anyone who can assemble a useful list of pointers to interesting documents, and find a market for that list.

This, I argue, is inevitable. The alternative is not that information remain expensive and bundled, but that copyright become meaningless. Digital storage of information is cheaper than the paper it replaces, and your published works will be photocopied and scanned-as long as it remains overwhelmingly cost-effective to do so.

I have a clear preference between these alternatives. Authors and editors need to be compensated in order to produce quality work. A scheme in which compensation continues to flow to the creators of a piece of work is superior to one in which compensation is incommensurate with use. The remaining question is whether the technology and the will to preserve copyright will leap ahead of the technology and the will to subvert it, or not. I can't speak to matters of resolve, but I will speak to matters of technology: storage, scanning, and OCR technologies are running ahead of the pricing models for current schemes of electronic commerce. While electronic commerce will eventually realize the kinds of economies necessary to preserve copyright, it's not certain that it won't already be too late.

Therefore, it becomes interesting to look at techniques for accelerating the move to sub-penny transactions. There are a few approaches that can be followed.

First, we can look to aggregated service providers. CompuServe, America Online, and (soon) the Microsoft Network provide simple billing models for access to information. The main problem is that the information that they can pay for is information for which they have contracted, and information that they deliver. This makes it hard for smaller information providers to gain access to users, makes it hard for consumers to retain much privacy from their service provider, makes it hard for information providers to determine whether the level of use reported by the service provider is accurate, and makes it hard for consumers to tell whether they are being accurately billed. Do consumers and vendors care about the privacy and accuracy issues? Not so much, today, but a good example of inappropriate behavior by an aggregated service provider might make people care.

Or, we can look at Millicent for an alternative architecture that provides better accuracy guarantees and modest privacy guarantees, if you believe that your payment intermediary isn't snooping all of your network traffic.

(back to Table of Contents)

Millicent overview

To achieve privacy, control over expenses, guarantees of payment, and lots of other really good properties, it's hard to beat cash. A good cheap scheme for digital cash would solve all of the problems right away, except for the problems caused by anonymous transactions, which any FBI or NSA agent will be happy to explain to you.

The problem is, there is no good cheap scheme for digital cash. Any scheme for digital cash is going to boil down to a signed message from a bank that claims to be worth some amount. Checking the signature is going to be relatively easy (although not immediate), but that's insufficient to tell whether a piece of digital cash is valid. Unlike real cash, duplicating digital cash is cheap and easy; you just send the number a second time to a different merchant. The bank can protect against double spending if the double-spender can be detected and assessed for the amount of duplication; this isn't going to be enough to prevent largely anonymous users from spending the same electronic coin several thousand times, or from skipping out on their debts. In such cases, the bank would have to deny payment to merchants, or make good on such fraud, driving the transaction price up. To protect against this, merchants could check each coin with the bank, to make sure that it hasn't been previously spent. This is going to add latency (due to queueing delays) and expense to a transaction system.

One way to make duplicate checking cheaper is to allow everyone to do it. If a local check on a serial number sufficed, we wouldn't have added much latency to the system. But we can't do this with cash; the idea of cash is that it's universal, and thus spendable at multiple vendors simultaneously. The first thing we'll do in Millicent is to change this.

Let's define a new kind of currency, which we'll call *scrip*. Scrip is like cash, in that it has some intrinsic value, but different in that it has that value only when spent with a specific merchant. Like electronic cash, scrip will consist of a signed message attesting that a particular serial number holds a particular value. Unlike typical cash, the message will contain an expiration date, and a particular vendor with whom the scrip can be redeemed.

Since we're concerned with providing transactions in the small, we'll allow ourselves some other simplifications. Ordinarily, digital cash is signed in some way that permits easy verification of the authenticity of a coin. Since a piece of scrip is of value only to its creator, and since the value of a piece of scrip is small, we don't need to be able to prove to anyone other than that vendor that a piece of scrip is authentic. We can assume that vendors are, in some way to be described later, controlled, and that consumers don't buy scrip from vendors that have cheated them before; thus, a consumer's risk in holding scrip from a vendor is limited to the face value of that scrip.

The best analogies to scrip are transit-system fare cards, pre-paid phone cards, and manufacturer coupons. A piece of scrip represents pre-paid value, whose authenticity is of interest only to the vendor, as long as the consumer is confident that it came from a trustworthy source. The vendor can employ whatever technology is appropriate to ensure authenticity and non-duplication.

In the scheme we propose for Millicent, the authenticity is guaranteed by a secret key. Since we don't need to prove the value to anyone other than the single merchant, it suffices to compute a signature incorporating a special secret. We propose that computing MD-5 of the plain text of the scrip followed by a secret key is sufficient, although any one-way hash function will suffice. Encryption would also suffice, but would be slower and might give us more trouble exporting the system.

To avoid duplication, the vendor will keep bit vectors corresponding to subranges of the issued serial numbers for scrip. As scrip expires (or becomes fully spent) the bit vectors can be discarded. This allows us to limit the storage requirements for a piece of scrip to an amortized 1+ bits. This permits the vendor to keep the database of valid scrip in main memory, speeding up transactions.

This leaves us with one big problem: buying scrip for a particular vendor. We can't buy it directly from the vendor, because that would commit us to large accounts with vendors, tying consumers down to large information providers. To get around this, we postulate scrip brokers: agents who buy contracts from vendors to produce scrip.

(back to Table of Contents)

Millicent scrip

Scrip in Millicent has to be unforgeable, unusable if copied, and inexpensive to create and validate. By using message digest functions as our primary form of encryption, and using shared-key cryptography when cryptography is necessary, we attempt to satisfy these requirements. A piece of scrip in Millicent consists of several parts. The scrip must identify the vendor, the expiration date, the value, and a small amount of information about the consumer. This information is intended to be limited to just the information necessary to satisfy requirements of age, location of residence (for sales tax and tariff purposes), and information needed to qualify for certain discounts, such as student status. Even so, some of this information may be sensitive for certain consumers; thus, we want to make sure that any portion of the data that the user wants to transmit only in encrypted form can be so transmitted.

For administrative purposes, and purposes of making change easily, it is desirable to create session keys. For this purpose, the first two parts of a piece of scrip are a session id and a session key. The session id serves to remind the vendor how to compute the session key. We propose that the session id contain in it the expiration date and a unique session number; if the vendor has a mapping from ranges of session numbers to secrets, the session key is just the result of signing the session id using the corresponding secret.

Let's introduce some notation. H will be a message-digest function, probably MD-5. SessionID will be the string corresponding to the session id; it includes N, the session number. The vendor remembers secrets X(N), where the mapping depends only on, say, the high-order bits of N, and remembers them as long as the expiration period for all sessions that have those high-order bits lasts.

Then the session key, k, is just H(SessionID^X(N)), where ^ denotes concatenation; this is what we mean by "signing a string (in this case, SessionID) with X(N)".

The remaining session information contains the vendor and consumer information. This information we denote by Info; when transmitting Info, we allow the consumer and vendor to decompose Info into encrypted and unencrypted pieces however it suits them, so long as Info can be recovered intact. When encrypting, the encryption should use k as the key.

Finally, we have to encode a coin. A coin, Coin, is a string containing a value, and a serial number Z. Additionally, we compute a coin key, j, a signature witnessing that Coin belongs to the current session. This signature is the result of computing H(SessionID^Info^Coin^X(Z)). For coins minted by a broker, we require that Z=N; we strongly recommend that X(Z) equal X(N) only when Z=N. That is, the same secret applies to a range of Z or N; we recommend that Z and N not be chosen in the same range except when equal, and that vendors choose distinct secrets for distinct ranges.

When a consumer wants to spend a piece of scrip, she sends SessionID, Info, Coin, and j to the vendor together with a request Req, and signs SessionID^Info^Coin^j^Req with k. Note that Info and Req may be sent partially encrypted under k; Req may contain instructions requesting that parts of the reply be encrypted. If any change is due to the consumer, the vendor sends new values for Coin and j back to the consumer along with the results of the request; to allow the consumer to check authenticity of the response, the whole response should be signed with k. Note that the new value for Coin will have a fresh serial number, and a (typically) decremented value.

Since H is a cryptographically strong message digest function, no one can create valid scrip without knowing X(Z) and X(N). In particular, no one can determine a session key without X(N), and no consumer can successfully find the matching value for a coin key after modifying SessionID, Info, or Coin. Thus, only parties knowing X(Z) (i.e., the vendor and broker licensed to create scrip for a subrange) can set Info and Coin; no one other than the vendor can change the monetary value of a piece of scrip or tamper with the consumer information. By choosing X(Z) different from X(N), the vendor prevents the broker from upgrading the value of consumer-held scrip; the vendor can now verify that all scrip for which X(Z) is a shared secret has Z=N, and the coin contains the value agreed to in the contract between the vendor and the broker.

Vendors may want to encrypt portions of their replies to requests, to prevent announcing their proprietary data to the world. For data which is "nearly public", that is, for data where the consumer has not requested privacy, it may suffice to encrypt the data with a single key for all users, and sell consumers that key encrypted by the session key.

We propose that for transmission of keys, it may be more efficient, and sufficiently secure, to use H and k as a pseudo one time pad. That is, by choosing and transmitting a random string, t, H(t^k) can be exclusive or-ed with a piece of data of the same length as the result in order to securely restrict transmission of that data to parties that know k. We particularly see an application of this to brokers, who need to transmit values for SessionID, Info, Coin, j, and k to consumers who have just bought scrip; of these, only k and some pieces of Info need to travel securely. The secure pieces of Info can be encrypted under k; the transmission of k must be accomplished securely, or else anyone observing that transmission could spend the scrip. Our current prototype implementation uses the one-time pad technique for encrypting k.

Validation of scrip is straightforward: SessionID is used to recover k, by signing SessionID with X(N);

since N is part of SessionID, this can be found. Info, if partially encrypted, is reconstructed using k, and SessionID, Info, Coin, Req and j are signed using k and matched against the transmitted value. A match proves that the sender either knew k, or retransmitted an old request, which would therefore be using a previously spent serial number. Additionally, SessionID, Info, and Coin are checked for tampering by comparing j to the signature using X(Z) of SessionID, Info, and Coin. Finally, Z is checked against the bit vector for freshness; if it is not fresh, this is either a replay, or an attempt by the consumer to respend used scrip.

(back to Table of Contents)

Money

To make the system work, actual money needs to change hands sometimes, and real signatures are needed to insure that these transfers are authorized and correspond to desired consequences.

To buy vendor scrip from a broker, the consumer needs some broker scrip. The consumer buys broker scrip by using some higher-priced form of electronic commerce (DigiCash, NetBill, the Visa/Microsoft protocol, SSL, whatever), which hopefully provides the ability to transmit k securely. After that, the consumer trades broker scrip for vendor scrip as needed, typically receiving lots of change, and a smaller quantity of vendor scrip. As mentioned above, this vendor scrip will have its own values for SessionID, Info, Coin, j, and k; at least k must be transmitted securely. We anticipate that consumers (or their browsers) will tune the policies of their software to buy enough scrip to cover some modest but anticipated number of transactions with the vendor; quantity discounts from the broker (or high markup on small purchases) will encourage scrip purchasers in this direction. This helps limit the transaction rate seen by brokers to an acceptable level. We anticipate that typical purchases will be in the range of five to ten dollars when buying broker scrip, and approximately one cent when buying vendor scrip.

In order to sell vendor scrip, the broker needs to acquire a license to produce scrip. The license needs to be enforceable through normal business practices; we expect that both pre-paid and consignment sale of scrip licenses will occur, with the choice being made by the broker based on past sales volumes and experience. We expect the development of a standard license for small-time information vendors; we anticipate that this will be a consignment sale, and that there will be a conventional discount to brokers for this scrip.

A license consists of a range of N, the secret X(N) for that range, an expiration period, an agreement as to what information Info should contain, and either a fixed value for each piece of scrip in the range or a maximum total value for all broker-produced scrip in the range and a value bound for individual pieces of scrip. As the system scales, it may become necessary for brokers to refer consumers to other brokers to satisfy scrip requests; in that case, the consumer's broker may offer to establish a small account with the vendor's broker for purposes of completing this transaction.

(back to Table of Contents)

Rekeying, refreshing, and redemption

In order to protect consumer information requests from the broker, consumers may want to rekey their scrip from a vendor using public-key technology. Vendors can be expected to accommodate this request, at a small price; all that's involved is choosing a new value for N, in a range not licensed to any broker, recomputing j and k, and transmitting the new SessionID, Info, Coin, j, and k, with the new k traveling encrypted. The encryption can either be an exclusive-or against a random string provided to the vendor under the vendor's public key, or encryption under a public key specified by the user. Unfortunately, this last is susceptible to a man-in-the-middle attack by the broker; since the only reason to rekey is to divorce your requests from observation by the broker, this may be inadequate, depending on the threat model we ascribe to our broker. The former has the disadvantage that it places the heavier computational load on the vendor, which doesn't scale as well. However, it is immune to man-in-the-middle attack.

Refreshing is similar, except that there's no requirement that the new k be strongly encrypted; the total value of the remaining scrip is decreasing, so there's less reason to be concerned about chaining keys than is typical in cryptographic protocols. We anticipate that refreshing soon-to-expire scrip will be a low-cost operation; vendors may, in fact, prefer to garbage collect subranges to eliminate the storage of bit vectors, and thus may be willing to refresh for free.

At some point, a browser may determine that certain scrip is unlikely to ever be of use. At this time, the consumer might like to redeem the vendor scrip for broker scrip. As part of a broker's license for vendor scrip, vendors receive a small license for generating broker scrip in order to redeem unused scrip. This scrip can either be spent by the consumer in the normal way, or merged with the consumer's existing broker scrip. It's likely that there will be a stiffer fee for these actions, to limit traffic and encourage careful heuristic design for scrip purchase and return.

(back to Table of Contents)

Problems

We've seen that consumers can't easily cheat in this system: if H is strong, consumers can't create scrip, they can't alter scrip, and they can't respend scrip. But who's to blame when something does go wrong? Vendors would be well-advised to anticipate that a certain fraction of transactions will get dropped at an awkward moment; the consumer's computer or network connectivity will fail, and the consumer both won't get what she wanted, and won't receive the change owed to her. To ameliorate this, vendors should keep an in-memory log of hash values for recent transactions, and generally allow replay of any transaction, or at least be willing to return the change from that transaction if the information previously delivered is now uselessly stale, as long as the request exactly matches the previous request.

If a consumer apparently tries to respend scrip, there are a few possible reasons:

1) the consumer could be cheating-this is what we're trying to catch.

2) the consumer could be confused-by permitting replay, we solve some of this; by writing browsers that are careful, we avoid a lot of this.

3) the broker could be selling the consumer used or bogus scrip-this is a problem that needs to be monitored, making sure that the vendor keeps track of whether clients of particular brokers have more trouble than others. Consumers who have done nothing wrong should insist that their broker make good on rejected scrip; if the broker won't, find another broker.

4) the vendor could be cheating by reporting valid, unspent scrip as bogus or spent-this is the flip side of problem 3, and one that brokers need to be on the lookout for. We anticipate that most brokers will

refund scrip to the consumer, until they develop a suspicion of either the vendor or the consumer. If the former, the broker can warn customers wanting vendor scrip of apparent problems with the vendor, demand better licensing terms, or stop doing business with the vendor. If the latter, the broker stops doing business with the customer. Consumers can't tell the difference between 3 and 4, except that their broker will be more helpful in case 4.

5) the cryptographic system in use could be insufficiently strong, or the security of the computers involved in producing or holding secrets may be inadequate. Stay vigilant.

(back to Table of Contents)

Performance

In the scheme outlined above, if no data is encrypted, the cost of a normal purchase is 3 hash functions for validation, and 1 to make change, plus the table lookups. Buying scrip from a broker is similar, plus the scrip generation costs: 2 hash functions to produce j and k, and one more using the suggested one-time pad encryption method for keys. Given current MD-5 speeds, this should permit transaction rates in the multiple hundreds per second. There are on the order of 3*10^8 seconds per year, so we should be able to support around 10^11 transactions per year. We'll divide by a factor of 10 to allow for idle time; this still leaves 10^10 transactions per year (contrastingly, a system involving a single public-key encryption per transaction using 768-bit keys will be lucky to reach 10^9 transactions). If each transaction at a broker is worth 10^-4 dollars (which is one percent of a penny transaction), the gross margin (I think that's the right term) is still 10^6 dollars per year, which should be enough to pay for the loaded cost of one computer. The transactions at vendors are much more profitable, although the load might be lower, and the transaction size is much smaller. Still, even at 10^-3 dollars per transaction, 10^9 transactions pays for the hardware and a person to keep it running; 10^10 transactions makes a lot of money for someone.

(back to Table of Contents)

Extensions

By adding more information to Coin, vendors can use scrip as a mechanism to deliver coupons: you bought the first page of this article, so here's some scrip that buys you the second page at half price, if you use it in the next hour.

By using the techniques of scrip without brokers, even vendors that prefer a subscription model can prevent subscribers from allowing unrestricted use of accounts by others. Once someone gives away their scrip, they can't use it any more, because the serial number won't match. Thus, at the very least, this solves the problem of someone at a university posting their access code for subscription to a service for all to use, without requiring the imposition of severe rate limiting for subscribers.

As computers get faster, we can extend Millicent-like techniques into lower-priced service domains. At around the time that conventional techniques begin to be able to handle the problems of by-the-page information purchase, we'll be ready to handle the problems of paying for bandwidth by the hop, or paying for electronic mail forwarding. At least, we'll be able to handle these problems for source-routed networks.

(back to Table of Contents)

Conclusions

The big new idea here is that cryptography can be useful to send yourself a message via an untrusted intermediary. Pushing the cost of storing infrequently-accessed information off to the client allows the servers to run quickly: the network acts as a perfect prefetch unit for loading your memory with the necessary state for the next operation. By way of example, a syndicated horoscope column may have tens of millions of readers; a server that had to store all of this information about subscribers might buckle under the seek time of accessing subscriber records. Pushing it off to clients allows us to have the salient parts of the database arrive in memory precisely when needed, at the small cost of verifying that the information hasn't been tampered with.

The secondary idea is that a service can help aggregate the transactions into a grain large enough to be acceptable to more conventional transaction handlers, without providing complete information about the transaction to the service. By making fund transfer explicit in each retrieval, all parties involved can instantly tell when they're being cheated; by limiting the scale of transaction, consumers can control the level of risk they accept.

(back to Table of Contents)

Acknowledgments

In spirit, this work clearly owes great debts to lots of previous work in this field. Kerberos was certainly inspirational, as were efforts by DigiCash and First Virtual.

(back to Table of Contents)

PayWord and MicroMint:
Two simple micropayment schemes

Ronald L. Rivest* and Adi Shamir**

May 7, 1996

*MIT Laboratory for Computer Science
545 Technology Square, Cambridge, Mass. 02139

**Weizmann Institute of Science
Applied Mathematics Department
Rehovot, Israel

{rivest,shamir}@theory.lcs.mit.edu

1 Introduction

We present two simple micropayment schemes, "PayWord" and "MicroMint," for making small purchases over the Internet. We were inspired to work on this problem by DEC's "Millicent" scheme[10]. Surveys of some electronic payment schemes can be found in Hallam-Baker [6], Schneier[16], and Wayner[18].

Our main goal is to minimize the number of public-key operations required per payment, using hash operations instead whenever possible. As a rough guide, hash functions are about 100 times faster than RSA signature verification, and about 10,000 times faster than RSA signature generation: on a typical workstation, one can sign two messages per second, verify 200 signatures per second, and compute 20,000 hash function values per second.

To support micropayments, exceptional efficiency is required, otherwise the cost of the mechanism will exceed the value of the payments. As a consequence, our micropayment schemes are light-weight compared to full macropayment schemes. We "don't sweat the small stuff": a user who loses a micropayment is similar to someone who loses a nickel in a candy machine. Similarly, candy machines aren't built with expensive mechanisms for detecting forged coins, and yet they work well in practice, and the overall level of abuse is low. Large-scale and/or persistent fraud must be detected and eliminated, but if the scheme delivers a volume of payments to the right parties that is roughly correct, we're happy.

In our schemes the players are brokers, users, and vendors. Brokers authorize users to make micropayments to vendors, and redeem the payments collected by the vendors. While user-vendor relationships are transient, broker-user and broker-vendor relationships are long-term. In a typical transaction a vendor sells access to a World-Wide Web page for one cent. Since a user may access only a few pages before moving on, standard credit-card arrangements incur unacceptably high overheads.

The first scheme, "PayWord," is a credit-based scheme, based on chains of "paywords" (hash values). Similar chains have been previously proposed for different purposes: by Lamport [9] and Haller (in S/Key) for access control [7], and by Winternitz [11] as a one-time signature scheme. The application of this idea for micropayments has also been independently discovered by Anderson et al. [2] and by Pederson [14], as we learned after distributing the initial draft of this paper. We discuss these related proposals further in Section 5. The user authenticates a complete chain to the vendor with a single public-key signature, and then successively reveals each payword in the chain to the vendor to make micropayments. The incremental cost of a payment is thus one hash function computation per party. PayWord is optimized for sequences of micropayments, but is secure and flexible enough to support larger variable-value payments as well.

The second scheme, "MicroMint," was designed to eliminate public-key operations altogether. It has lower security but higher speed. It introduces a new paradigm of representing coins by k-way hash-function collisions. Just as for a real mint, a broker's "economy of scale" allows him to produce large quantities of such coins at very low cost per coin, while small-scale forgery attempts can only produce coins at a cost exceeding their value.

2 Generalities and Notation

We use public-key cryptography (e.g. RSA with a short public exponent). The public keys of the broker B, user U, and vendor V are denoted PK_B, PK_U, and PK_V, respectively; their secret keys are denoted SK_B, SK_U, and SK_V. A message M with its digital signature produced by secret key SK is denoted $\{M\}_{SK}$. This signature can be verified using the corresponding public key PK.

We let h denote a cryptographically strong hash function, such as MD5[15] or SHA[13]. The output (nominally 128 or 160 bits) may be truncated to shorter lengths as described later. The important property of h is its one-wayness and collision-resistance; a very large search should be required to find a single input producing a given output, or to find two inputs producing the same output. The input length may, in some cases, be equal to the output length.

3 PayWord

PayWord is credit-based. The user establishes an account with a broker, who issues her a digitally-signed PayWord Certificate containing the broker's name, the user's name and IP-address, the user's public key, the expiration date, and other information. The certificate has to be renewed by the broker (e.g. monthly), who will do so if the user's account is in good standing. This certificate authorizes the user to make Payword chains, and assures vendors that the user's paywords are redeemable by the broker. We assume in this paper that each payword is worth exactly one cent (this could be varied).

In our typical application, when U clicks on a link to a vendor V's non-free web page, his browser determines whether this is the first request to V that day. For a first request, U computes and signs a "commitment" to a new user-specific and vendor-specific chain of paywords w_1, w_2, …, w_n. The user creates the payword chain in reverse order by picking the last payword w_n at random, and then computing

$$w_i = h(w_{i+1})$$

for $i = n-1, n-2, \ldots, 0$. Here w_0 is the root of the payword chain, and is not a payword itself. The commitment contains the root w_0, but not any payword w_i for $i > 0$. Then U provides this commitment and her certificate to V, who verifies their signatures.

The i-th payment (for $i = 1, 2, \ldots$) from U to V consists of the pair (w_i, i), which the vendor can verify using w_{i-1}. Each such payment requires no calculations by U, and only a single hash operation by V.

At the end of each day, V reports to B the last (highest-indexed) payment (w_l, l) received from each user that day, together with each corresponding commitment. B charges U's account l cents and pays l cents into V's account. (The broker might also charge subscription and/or transaction fees, which we ignore here.)

A fundamental design goal of PayWord is to minimize communication (particularly on-line communication) with the broker. We imagine that there will be only a few nationwide brokers; to prevent them from becoming a bottleneck, it is important that their computational burden be both reasonable and "off-line." PayWord is an "off-line" scheme: V does not need to interact with B when U first contacts V, nor does V need to interact with B as each payment is made. Note that B does not even receive every payword spent, but only the *last* payword spent by each user each day at each vendor.

PayWord is thus extremely efficient when a user makes repeated requests from the same vendor, but is quite effective in any case. The public-key operations required by V are only signature verifications, which are relatively efficient. We note that Shamir's probabilistic signature screening techniques[17] can be used here to reduce the computational load on the vendor even further. Another application where PayWord is well-suited is the purchase of pay-per-view movies; the user can pay a few cents for each minute of viewing time.

This completes our overview; we now give some technical details.

3.1 User-Broker relationship and certificates

User U begins a relationship with broker B by requesting an account and a PayWord Certificate. She gives B over a secure authenticated channel: her credit-card number, her public key PK_U, and her "delivery address" A_U. Her aggregated PayWord charges will be charged to her credit-card account. Her delivery address is her Internet/email or her U.S. mail address; her certificate will only authorize payments by U for purchases to be delivered to A_U.

The user's certificate has an expiration date E. Certificates might expire monthly, for example. Users who don't pay their bills won't be issued new certificates.

The broker may also give other (possibly user-specific) information I_U in the certificate, such as: a certificate serial number, credit limits to be applied per vendor, information on how to contact the broker, broker/vendor terms and conditions, etc.

The user's certificate C_U thus has the form:

$$C_U = \{B, U, A_U, PK_U, E, I_U\}_{SK_B} .$$

The PayWord certificate is a statement by B to any vendor that B will redeem authentic paywords produced by U turned in before the given expiration date (plus a day's grace).

PayWord is not intended to provide user anonymity. Although certificates could contain user account numbers instead of user names, the inclusion of A_U effectively destroys U's anonymity. However, some privacy is provided, since there is no record kept as to which documents were purchased.

If U loses her secret key she should report it at once to B. Her liability should be limited in such cases, as it is for credit-card loss. However, if she does so repeatedly the broker may refuse her further service. The broker may also keep a "hot list" of certificates whose users have reported lost keys, or which are otherwise problematic.

As an alternative to hot-lists, one can use hash-chains in a different manner as proposed by Micali [12] to provide daily authentication of the user's certificate. The user's certificate would additionally contain the root w'_0 of a hash chain of length 31. On day $j-1$ of the month, the broker will send the user (e.g. via email) the value w'_j if and only if the user's account is still in good standing. Vendors will then demand of each user the appropriate w' value before accepting payment.

3.2 User-Vendor relationships and payments

User-vendor relationships are transient. A user may visit a web site, purchase ten pages, and then move on elsewhere.

Commitments

When U is about to contact a new vendor V, she computes a fresh payword chain w_1, …, w_n with root w_0. Here n is chosen at the user's convenience; it could be ten or ten thousand. She then computes her commitment for that chain:

$$M = \{V, C_U, w_0, D, I_M\}_{SK_U} \ .$$

Here V identifies the vendor, C_U is U's certificate, w_0 is the root of the payword chain, D is the current date, and I_M is any additional information that may be desired (such as the length n of the payword chain). M is signed by U and given to V. (Since this signature is necessarily "on-line," as it contains the vendor's name, the user might consider using an "on-line/off-line" signature scheme[5].)

This commitment authorizes B to pay V for any of the paywords w_1, …, w_n that V redeems with B before date D (plus a day's grace). Note that paywords are *vendor-specific* and *user-specific*; they are of no value to another vendor.

Note that U must sign a commitment for each vendor she pays. If she rapidly switches between vendors, the cost of doing so may become noticeable. However, this is PayWord's only significant computational requirement, and the security it provides makes PayWord usable even for larger "macropayments" (e.g. software selling at \$19.99).

The vendor verifies U's signature on M and the broker's signature on C_U (contained within M), and checks expiration dates.

The vendor V should cache verified commitments until they expire at the end of the day. Otherwise, if he redeemed (and forgot) paywords received before the expiration date of the commitment, U could cheat V by replaying earlier commitments and paywords. (Actually, to defeat this attack, V need store only a short hash of each commitment he has reported to B already today.)

The user should preferably also cache her commitment until she believes that she is finished ordering information from V, or until the commitment expires. She can always generate a fresh commitment if she re-visits a vendor whose commitment she has deleted.

Payments

The user and vendor need to agree on the amount to be paid. In our exemplary application, the price of a web page is typically one cent, but could be some other amount. A web page should presumably be free if the user has already purchased it that day, and is just requesting it again because it was flushed from his cache of pages.

A payment P from U to V consists of a payword and its index:

$$P = (w_i, i) \ .$$

The payment is short: only twenty or thirty bytes long. (The first payment to V that day would normally accompany U's corresponding commitment; later payments are just the payword and its index, unless the previous chain is exhausted and a new chain must be committed to.) The payment is not signed by U, since it is self-authenticating (using the commitment).

The user spends her paywords in order: w_1 first, then w_2, and so on. If each payword is worth one cent, and each web page costs one cent, then she discloses w_i to V when she orders her i-th web page from V that day.

This leads to the PayWord payment policy: *for each commitment a vendor V is paid l cents, where (w_l, l) is the corresponding payment received with the largest index.* This means that V needs to store only one payment from each user: the one with the highest index. Once a user spends w_i, she can not spend w_j for $j < i$. The broker can confirm the value to be paid for w_l by determining how many applications of h are required to map w_l into w_0.

PayWord supports variable-size payments in a simple and natural manner. If U skips paywords, and gives w_7 after giving w_2, she is giving V a nickel instead of a penny. When U skips paywords, during verification V need only apply h a number of times proportional to the value of the payment made.

A payment does not specify what item it is payment for. The vendor may cheat U by sending him nothing, or the wrong item, in return. The user bears the risk of losing the payment, just as if he had put a penny in the mail. Vendors who so cheat their customers will be shunned. This risk can be moved to V, if V specifies payment *after* the document has been delivered. If U doesn't pay, V can notify B and/or refuse U further service. For micropayments, users and vendors might find either approach workable.

3.3 Vendor-Broker relationships and redemption

A vendor V needn't have a prior relationship with B, but does need to obtain PK_B in an authenticated manner, so he can authenticate certificates signed by B. He also needs to establish a way for B to pay V for paywords redeemed. (Brokers pay vendors by means outside the PayWord system.)

At the end of each day (or other suitable period), V sends B a redemption message giving, for each of B's users who have paid V that day (1) the commitment C_U received from U, (2) the last payment $P = (w_l, l)$ received from U.

The broker then needs to (1) verify each commitment received (he only needs to verify user signatures, since he can recognize his own certificates), including checking of dates, etc., and (2) verify each payment (w_l, l) (this requires l hash function applications). We assume that B normally honors all valid redemption requests.

Since hash function computations are cheap, and signature verifications are only moderately expensive, B's computational burden should be reasonable, particularly since it is more-or-less proportional to the payment volume he is supporting; B can charge transaction or subscription fees adequate to cover his computation costs. We also note that B never needs to respond in real-time; he can batch up his computations and perform them off-line overnight.

3.4 Efficiency

We summarize PayWord's computational and storage requirements:

- The broker needs to sign each user certificate, verify each user commitment, and perform one hash function application per payment. (All these computations are off-line.) The broker stores copies of user certificates and maintains accounts for users and vendors.
- The user needs to verify his certificates, sign each of his commitments, and perform one hash function application per payword committed to. (Only signing commitments is an on-line computation.) He needs to store his secret key SK_U, his active commitments, the corresponding payword chains, and his current position in each chain.
- The vendor verifies all certificates and commitments received, and performs one hash function application per payword received or skipped over. (All his computations are on-line.) The vendor needs to store all commitments and the last payment received per commitment each day.

3.5 Variations and Extensions

In one variation, $h(\cdot)$ is replaced by $h_s(\cdot) = h(s, \cdot)$, where s is a "salt" (random value) specified in the commitment. Salting may enable the use of faster hash functions or hash functions with a shorter output length (perhaps as short as 64–80 bits).

The value of each payword might be fixed at one cent, or might be specified in C_U or M. In a variation, M might authenticate several chains, whose paywords have different values (for penny paywords, nickel paywords, etc.).

The user name may also need to be specified in a payment if it is not clear from context. If U has more than one payword chain authorized for V, then the payment should specify which is relevant.

Paywords could be sold on a debit basis, rather than a credit basis, but only if the user interacts with the broker to produce each commitment: the certificate could require that the broker, rather than the user, sign each commitment. The broker can automatically refund the user for unused paywords, once the vendor has redeemed the paywords given to him.

In some cases, for macropayments, it might be useful to have the "commitment" act like an electronic credit card order or check without paywords being used at all. The commitment would specify the vendor and the amount to be paid.

The broker may specify in user certificates other terms and conditions to limit his risk. For example, B may limit the amount that U can spend per day at any vendor. Or, B may refuse payment if U's name is on B's "hot list" at the beginning of the day. (Vendors can down-load B's hot-list each morning.) Or, B may refuse to pay if U's total expenditures over all vendors exceeds a specified limit per day. This protects B from extensive liability if SK_U is stolen and abused. (Although again, since C_U only authorizes delivery to A_U, risk is reduced.) In these cases vendors share the risk with B.

Instead of using payword chains, another method we considered for improving efficiency was to have V *probabilistically* select payments for redemption. We couldn't make this idea work out, and leave this approach as an open problem.

4 MicroMint

MicroMint is designed to provide reasonable security at very low cost, and is optimized for unrelated low-value payments. MicroMint uses *no* public-key operations at all.

MicroMint "coins" are produced by a broker, who sells them to users. Users give these coins to vendors as payments. Vendors return coins to the broker in return for payment by other means.

A coin is a bit-string whose validity can be easily checked by anyone, but which is hard to produce. This is similar to the requirements for a public-key signature, whose complexity makes it an overkill for a transaction whose value is one cent. (PayWord uses signatures, but not on every transaction.)

MicroMint has the property that generating many coins is very much cheaper, per coin generated, than generating few coins. A large initial investment is required to generate the first coin, but then generating additional coins can be made progressively cheaper. This is similar to the economics for a regular mint, which invests in a lot of expensive machinery to make coins economically. (It makes no sense for a forger to produce coins in a way that costs more per coin produced than its value.)

The broker will typically issue new coins at the beginning of each month; the validity of these coins will expire at the end of the month. Unused coins are returned to the broker at the end of each month, and new coins can be purchased at the beginning of each month. Vendors can return the coins they collect to the broker at their convenience (e.g. at the end of each day).

We now describe the "basic" variant of MicroMint. Many extensions and variations are possible on this theme; we describe some of them in section 4.2.

Hash Function Collisions

MicroMint coins are represented by *hash function collisions*, for some specified one-way hash function h mapping m-bit strings x to n-bit strings y. We say that x is a pre-image of y if $h(x) = y$. A pair of distinct m-bit strings (x_1, x_2) is called a *(2-way) collision* if $h(x_1) = h(x_2) = y$, for some n-bit string y.

If h acts "randomly," the only way to produce even one acceptable 2-way collision is to hash about $\sqrt{2^n} = 2^{n/2}$ x-values and search for repeated outputs. This is essentially the "birthday paradox." (We ignore small constants in our analyses.)

Hashing c times as many x-values as are needed to produce the first collision results in approximately c^2 as many collisions, for $1 \leq c \leq 2^{n/2}$, so producing collisions can be done increasingly efficiently, per coin generated, once the threshold for finding collisions has been passed.

Coins as k-way collisions

A problem with 2-way collisions is that choosing a value of n small enough to make the broker's work feasible results in a situation where coins can be forged a bit too easily by an adversary. To raise the threshold further against would-be forgers, we propose using k-way collisions instead of 2-way collisions.

A k-way collision is a set of k distinct x-values $x_1, x_2, \ldots, x_k$ that have the same hash value y. The number of x-values that must be examined before one expects to see the first k-way collision is then approximately $2^{n(k-1)/k}$. If one examines c times this many x-values, for $1 \le c \le 2^{n/k}$, one expects to see about c^k k-way collisions. Choosing $k > 2$ has the dual effect of delaying the threshold where the first collision is seen, and also accelerating the rate of collision generation, once the threshold is passed.

We thus let a k-way collision $(x_1, \ldots, x_k)$ represent a coin. The validity of this coin can be easily verified by anyone by checking that the x_i's are distinct and that

$$h(x_1) = h(x_2) = \cdots = h(x_k) = y$$

for some n-string y.

Minting coins

The process of computing $h(x) = y$ is analogous to tossing a ball (x) at random into one of 2^n bins; the bin that ball x ends up in is the one with index y. A coin is thus a set of k balls that have been tossed into the same bin. Getting k balls into the same bin requires tossing a substantial number of balls altogether, since balls can not be "aimed" at a particular bin. To mint coins, the broker will create 2^n bins, toss approximately $k2^n$ balls, and create one coin from each bin that now contains at least k balls. With this choice of parameters each ball has a chance of roughly 1/2 of being part of a coin.

Whenever one of the 2^n bins has k or more balls in it, k of those balls can be extracted to form a coin. Note that if a bin has more than k balls in it, the broker can in principle extract k-subsets in multiple ways to produce several coins. However, an adversary who obtains two different coins from the same bin could combine them to produce multiple new coins. Therefore, we recommend that *a MicroMint broker should produce at most one coin from each bin.* Following this rule also simplifies the Broker's task of detecting multiply-spent coins, since he needs to allocate a table of only 2^n bits to indicate whether a coin with a particular n-bit hash value has already been redeemed.

A small problem in this basic picture, however, is that computation is much cheaper than storage. The number of balls that can be tossed into bins in a month-long computation far exceeds both the number of balls that can be memorized on a reasonable number of hard disks and the number of coins that the broker might realistically need to mint. One could attempt to balance the computation and memory requirements by utilizing a very slow hash algorithm, such as DES iterated many times. Unfortunately, this approach also slows down the verification process.

A better approach, which we adopt, is to make most balls unusable for the purpose of minting coins. To do so, we say that a ball is "good" if the high-order bits of the hash value y have a value z specified by the broker. More precisely, let $n = t + u$ for some specified nonnegative integers t and u. If the high-order t bits of y are equal to the specified value z then the value y is called "good, " and the low-order u bits of y determine the index of the bin into which the (good) ball x is tossed. (General x values are referred to merely as "balls," and those that are not good can be thought of as having been conceptually tossed into nonexistent virtual bins that are "out of range.")

A proper choice of t enables us to balance the computational and storage requirements of the broker, without slowing down the verification process. It slows down the generation process by a factor of 2^t, while limiting the storage requirements of the broker to a small multiple of the number of coins to be generated. The broker thus tosses approximately $k2^n$ balls, memorizes about $k2^u$ good balls that he tosses into the 2^u bins, and generates from them approximately $(1/2) \cdot 2^u$ valid coins.

Remark: We note that with standard hash functions, such as MD5 and DES, the number of ouput bits produced may exceed the number n of bits specified in the broker's parameters. A suitable hash function for the broker can be obtained by discarding all but the low-order n bits of the standard hash function output. This discarding of bits other than the low-order n bits is a different process than that of specifying a particular value for the high-order t bits out of the n that was described above.

A detailed scenario

Here is a detailed sketch of how a typical broker might proceed to choose parameters for his minting operating for a given month. The calculations are approximate (values are typically rounded to the nearest power of two), but instructive; they can be easily modified for other assumptions.

The broker will invest in substantial hardware that gives him a computational advantage over would-be forgers, and run this hardware continuously for a month to compute coins valid for the next month. This hardware is likely to include many special-purpose chips for computing h efficiently.

We suppose that the broker wishes to have a net profit of $1 million per month (approximately 2^{27} cents/month). He charges a brokerage fee of 10%. That is, for every coin worth one cent that he sells, he only gives the vendor 0.9 cents when it is redeemed. Thus, the broker needs to sell one billion coins per month (approximately 2^{30} coins/month) to collect his $1M fee. If an average user buys 2500 ($25.00) coins per month, he will need to have a customer base of 500,000 customers.

The broker chooses $k = 4$; a coin will be a good 4-way collision.

To create 2^{30} coins, the broker chooses $u = 31$, so that he creates an array of 2^{31} (approximately two billion) bins, each of which can hold up to 4 x-values that hash to an n-bit value that is the concantenation of a fixed t-bit pattern z and the u-bit index of the bin.

The broker will toss an average of 4 balls into each bin. That is, the broker will generate $4 \cdot 2^{31} = 2^{33}$ (approximately eight billion) x-values that produce good y-values. When he does so, the probability that a bin then contains 4 or more x-values (and thus can yield a coin) is about 1/2. (Using a Poisson approximation, it can be calculated that the correct value is approximately 0.56.) Since each of the 2^{31} bins produces a coin with probability 1/2, the number of coins produced is 2^{30}, as desired.

In order to maximize his advantage over an adversary who wishes to forge coins, the broker invests in special-purpose hardware that allows him to compute hash values very quickly. This will allow him to choose a relatively large value of t, so that good hash values are relatively rare. This increases the work factor for an adversary (and for the broker) by a factor of 2^t. The broker chooses his hash function h as the low-order n bits of the encryption of some fixed value v_0 with key x under the Data Encryption Standard (DES):

$$h(x) = [DES_x(v_0)]_{1 \ldots n} \, .$$

The broker purchases a number of field-programmable gate array (FPGA) chips, each of which is capable of hashing approximately 2^{25} (approximately 30 million) x-values per second. (See [3].) Each such chip costs about \$200; we estimate that the broker's actual cost per chip might be closer to \$400 per chip when engineering, support, and associated hardware are also considered. The broker purchases 2^8 (= 256) of these chips, which costs him about \$100,000. These chips can collectively hash 2^{33} (approximately 8.6 billion) values per second. Since there are roughly 2^{21} (two million) seconds in a month, they can hash about 2^{54} (approximately 18 million billion) values per month.

Based on these estimates the broker chooses $n = 52$ and $t = 21$ and runs his minting operation for one month. Of the $k2^n = 2^{54}$ hash values computed, only one in 2^{21} will be good, so that approximately 2^{33} good x-values are found, as necessary to produce 2^{30} coins.

Storing a good $(x, h(x))$ pair takes less than 16 bytes. The total storage required for all good pairs is less than 2^{37} bytes (128 Gigabytes). Using standard magnetic hard disk technology costing approximately \$300 per Gigabyte, the total cost for storage is less than \$40,000. The total cost for the broker's hardware is thus less than \$150,000, which is less than 15% of the first month's profit.

Rather than actually writing each pair into a randomly-accessible bin, the broker can write the 2^{33} good pairs sequentially to the disk array, and then sort them into increasing order by y value, to determine which are in the same bin. With a reasonable sorting algorithm, the sorting time should be under one day.

Selling coins

Towards the end of each month, the broker begins selling coins to users for the next month. At the beginning of each month, B reveals the new validity criterion for coins to be used that month. Such sales can either be on a debit basis or a credit basis, since B will be able to recognize coins when they are returned to him for redemption. In a typical purchase, a user might buy \$25.00 worth of coins (2500 coins), and charge the purchase to his credit card. The broker keeps a record of which coins each user bought. Unused coins are returned to the broker at the end of each month.

Making payments

Each time a user purchases a web page, he gives the vendor a previously unspent coin $(x_1, x_2, \ldots, x_k)$. (This might be handled automatically by the user's web browser when the user clicks on a link that has a declared fee.) The vendor verifies that it is indeed a good k-way collision by computing $h(x_i)$ for $1 \leq i \leq k$, and checking that the values are equal and good. Note that while the broker's minting process was intentionally slowed down by a factor of 2^t, the vendor's task of verifying a coin remains extremely efficient, requiring only k hash computations and a few comparisons (in our proposed scenario, $k = 4$).

Redemptions

The vendor returns the coins he has collected to the broker at the end of each day. The broker checks each coin to see if it has been previously returned, and if not, pays the vendor one cent (minus his brokerage fee) for each coin. We propose that if the broker receives a specific coin more than once, he does not pay more than once. Which vendor gets paid can be decided arbitrarily or randomly by the broker. This may penalize vendors, but eliminates any financial motivation a vendor might have had to cheat by redistributing coins he has collected to other vendors.

4.1 Security Properties

We distinguish between small-scale attacks and large-scale attacks. We believe that users and vendors will have little motivation to cheat in order to gain only a few cents; even if they do, the consequences are of no great concern. This is similar to the way ordinary change is handled: many people don't even bother to count their change following a purchase. Our security mechanisms are thus primarily designed to discourage large-scale attacks, such as massive forgery or persistent double-spending.

Forgery

Small-scale forgery is too expensive to be of interest to an adversary: with the recommended choice of $k = 4$, $n = 54$, and $u = 31$, the generation of the first forged coin requires about 2^{45} hash operations. Since a standard work-station can perform only 2^{14} hash operations per second, a typical user will need 2^{31} seconds (about 80 years) to generate just one forged coin on his workstation.

Large-scale forgery can be detected and countered as follows:

- All forged coins automatically become invalid at the end of the month.
- Forged coins can not be generated until after the broker announces the new monthly coin validity criterion at the beginning of the month.
- The use of hidden predicates (described below) gives a finer time resolution for rejecting forged coins without affecting the validity of legal coins already in circulation.
- The broker can detect the presence of a forger by noting when he receives coins correspondings to bins that he did not produce coins from. This works well in our scenario since only about half of the bins produce coins. To implement this the broker need only work with a bit-array having one bit per bin.
- The broker can at any time declare the current period to be over, recall all coins for the current period, and issue new coins using a new validation procedure.
- The broker can simultaneously generate coins for several future months in a longer computation, as described below; this makes it harder for a forger to catch up with the broker.

Theft of coins

If theft of coins is judged to be a problem during initial distribution to users or during redemption by vendors, it is easy to transmit coins in encrypted form during these operations.

User/broker and vendor/broker relationships are relatively stable, and long-term encryption keys can be arranged between them.

To protect coins as they are being transferred over the Internet from user to vendor, one can of course use public-key techniques to provide secure communication. However, in keeping with our desire to minimize or eliminate public-key operations, we propose below another mechanism, which makes coins user-specific. This does not require public-key cryptography, and makes it harder to re-use stolen coins.

Another concern is that two vendors may collude so that both attempt to redeem the same coins. The recommended solution is that a broker redeem a coin at most once, as discussed earlier. Since this may penalize honest vendors who receive stolen coins, we can make coins vendor-specific as well as user-specific, as described below.

Double-spending

Since the MicroMint scheme is not anonymous, the broker can detect a doubly-spent coin, and can identify which vendors he received the two instances from. He also knows which user the coin was issued to. With the vendors' honest cooperation, he can also identify which users spent each instance of that coin. Based on all this information, the broker can keep track of how many doubly-spent coins are asssociated with each user and vendor. A large-scale cheater (either user or vendor) can be identified by the large number of duplicate coins associated with his purchases or redemptions; the broker can then drop a large-scale cheater from the system. A small-scale cheater may be hard to identify, but, due to the low value of individual coins, it is not so important if he escapes identification.

MicroMint does not provide any mechanism for preventing purely malicious framing (with no financial benefit to the framer). We believe that the known mechanisms for protecting against such behavior are too cumbersome for a light-weight micropayment scheme. Since MicroMint does not use real digital signatures, it may be hard to legally prove who is guilty of duplicating coins. Thus, a broker will not be able to pursue a cheater in court, but can always drop a suspected cheater from the system.

4.2 Variations

User-specific coins

We describe two proposals for making coins that are user-specific in a way that can be easily checked by vendors. Such coins, if stolen, are of no value to most other users. This greatly reduces the motivation for theft of coins.

In the first proposal, the broker splits the users into "groups," and gives each user coins whose validity depends on the identity of the group. For example, the broker can give user U coins that satisfy the additional condition $h'(x_1, x_2, \ldots, x_k) = h'(U)$, where hash function h' produces short (e.g. 16-bit) output values that indicate U's group. A vendor can easily check this condition, and reject a coin that is not tendered by a member of the correct group.

The problem with this approach is that if the groups are too large, then a thief can easily find users of the appropriate group who might be willing to buy stolen coins. On the other hand, if the groups are too small (e.g. by placing each user is in his own group), the broker may be forced to precompute a large excess of coins, just to ensure that he has a large enough supply to satisfy each user's unpredictable needs.

In the second proposal, we generalize the notion of a "collision" to more complicated combinatorial structures. Formally, a coin $(x_1, \ldots, x_k)$ will be valid for a user U if the images $y_1 = h(x_1)$, $y_2 = h(x_2)$, $\ldots$, $y_k = h(x_k)$ satisfy the condition

$$y_{i+1} - y_i = d_i \pmod{2^u}$$

for $i = 1, 2, \ldots, k-1$, where

$$(d_1, d_2, \ldots, d_{k-1}) = h'(U)$$

for a suitable auxiliary hash function h'. (The original proposal for representing coins as collisions can be viewed as the special case where all the distances d_i's between the k bins are zero.)

To mint coins of this form, the broker fills up most of his bins by randomly tossing balls into them, except that now it is not necessary to have more than one ball per bin. We emphasize that this pre-computation is not user-specific, and the broker does not need to have any prior knowledge of the number of coins that will be requested by each user, since each good ball can be used in a coin for *any* user. After this lengthy pre-computation, the broker can quickly create a coin for any user U by

- Computing $(d_1, \ldots, d_{k-1}) = h'(U)$.
- Picking a random bin index y_1. (This bin should have been previously unused as a y_1 for another coin, so that y_1 can be used as the "identity" of the coin when the broker uses a bit-array to determine which coins have already been redeemed.)
- Computing $y_{i+1} = y_i + d_i \pmod{2^u}$ for $i = 1, 2, \ldots, k-1$,
- Taking a ball x_1 out of bin y_1, and taking a copy of one ball out of each bin y_2, $\ldots$, y_k. (If any bin y_i is empty, start over with a new y_1.) Note that balls may be re-used in this scheme.
- Producing the ordered k-tuple $(x_1, \ldots, x_k)$ as the output coin.

A convenient feature of this scheme is that it is easy to produce a large number of coins for a given user even when the broker's storage device is a magnetic disk with a relatively slow seek time. The idea is based on the observation that if the y_1 values for successive coins are consecutive, then so also will be the y_i values for each i, $1 < i \leq k$. Therefore, a request for 2500 new coins with $k = 4$ requires only four disk seeks, rather than 10,000 seeks: at 10 milliseconds per seek, this reduces the total seek time from 100 seconds to only 40 milliseconds.

Note that in principle coins produced for different users could re-use the same ball x_i. Conceivably, someone could forge a new coin by combining pieces of other coins he has seen. However, he is unlikely to achieve much success by this route unless he sees balls from a significant fraction of all the bins. For example, suppose that there are 2^{31} bins, of which the forger has seen a fraction 2^{-10} (i.e., he has collected 2^{21} balls from coins spent by other users). Then the expected number of coins he can piece together from these balls that satisfy the condition of being a good coin for himself is only $2^{31}(2^{-10})^3 = 2$. (Even if he had 1000 customers for these coins, he would expect to make only 2000 coins total, or two coins per customer on the average.) Thus, we are not too concerned about this sort of "cut-and-paste" forgery.

Vendor-specific coins

To further reduce the likelihood that coins will be stolen, the user can give coins to vendors in such a way that each coin can be redeemed only by a small fraction of the vendors. This technique makes a stolen coin less desirable, since it is unlikely to be accepted by a vendor other than the one where it was originally spent. The additional check of validity can be carried out both by the vendor and by the broker. (Having vendor-specific coins is also a major feature of the Millicent [10] scheme.)

The obvious difficulty is that neither the broker nor the user can predict ahead of time which vendors the user will patronize, and it is unreasonable to force the user to purchase in advance coins specific for each possible vendor. Millicent adopts the alternative strategy whereby the user must contact the broker in real-time whenever the user needs coins for a new vendor. (He also needs to contact the broker to return excess unused coins that are specific to that vendor.) We can overcome these problems with an extension of the user-specific scheme described above, in which the user purchases a block of "successive" MicroMint coins.

Intuitively, the idea is the following. Choose a value v (e.g. 1024) less than u. Let a u-bit bin-index y be divided into a $u-v$-bit upper part y' and a v-bit lower part y''. We consider that y' specifies a "superbin" index and that y'' specifies a bin within that superbin. A user now purchases balls in bulk and makes his own coins. He purchases balls by the superbin, obtaining 2^v balls per superbin with one ball in each bin of the superbin. He buy k superbins of balls for 2^v cents. A coin from user U is valid for redemption by vendor V if:

$$y'_{i+1} = y'_i + d'_i \pmod{2^{u-v}} \text{ for i} = 1, \ldots, \text{k} - 1,$$

and

$$y''_{i+1} = y''_i + d''_i \pmod{2^v} \text{ for i} = 1, \ldots, \text{k} - 1,$$

where

$$h'(U) = (d'_1, \ldots, d'_{k-1})$$

and

$$h''(V) = (d''_1, \ldots, d''_{k-1})\,.$$

The broker chooses the next available superbin as the first superbin to give the user; the other superbins are then uniquely determined by the differences $\{d'_i\}$ defined by the user's identity and the choice of the first superbin. Analogously, to make a coin for a particular vendor the user chooses a ball from the next bin from his first superbin, and must use balls from bins in the other superbins that are then uniquely determined by the differences $\{d''_i\}$ defined by the vendor's identity and the choice of the first bin. Note that balls from the first superbin are used only once, to permit detection of double-spending, whereas balls from the other superbins may appear more than once (in coins paid to different vendors), or not at all. It may be difficult for a broker to create superbins that are perfectly full even if he throws more balls. He might sell superbins that are almost full, but then a user may have difficulty producing some coins for some vendors. To compensate, the broker can reduce the price by one cent for each empty bin sold.

Simultaneously generating balls for multiple months

Our major line of defense against large-scale forgery is the fact that the broker can compute coins in advance, whereas a forgery attempt can only be started once the new validity condition for the current month is announced. We now describe a technique whereby computing the balls for a single month's coins takes eight months, but the broker doesn't fall behind because he can generate balls for eight future months concurrently. The forger will thus have the dual problems of starting late and being too slow, even if he uses the same computational resources as the real broker.

In this method, the broker changes the monthly validity criterion, not by changing the hash function h, but by announcing each month a new value z such that ball x is good when the high-order t bits of $h(x)$ are equal to z. The broker randomly and secretly chooses in advance the values z that will be used for each of the next eight months. Tossing a ball still means performing one hash function computation, but the tossed ball is potentially "good" for any of the next eight months, and it is trivial for the broker to determine if this is the case. In contrast, the forger only knows the current value of z, and can not afford to memorize all the balls he tosses, since memory is relatively expensive and only a tiny fraction (e.g., 2^{-21} in our running example) of the balls are considered "good" at any given month.

We now describe a convenient way of carrying out this calculation. Assume that at the beginning of the month j, the broker has all of the balls needed for month j, $7/8$ of the balls needed for month $j+1$, $6/8$ of the balls needed for month $j+2$, ..., and $1/8$ of the balls needed in for month $j+7$. During month j, the broker tosses balls by randomly picking x values, calculating $y = h(x)$, and checking whether the top-most t bits of y are equal to any of the z values to be used in months $j+1, \ldots, j+8$. To slow the rate at which he generates good balls for each upcoming month, he increases n and t each by three. After the month-long computation, we expect him to have all the coins he needs for month $j+1$, $7/8$ of the coins he needs for month $j+2$, and so on; this is the desired "steady-state" situation. The broker needs four times as much storage to hold the balls generated for future months, but balls for future months can be temporarily stored on inexpensive magnetic tapes because he doesn't need to respond quickly to user requests for those coins yet.

Hidden Predicates

The "hidden predicate" technique for defeating forgers works as follows. We choose $m > n$, and require each m-bit pre-image to satisfy a number of hidden predicates. The hidden predicates should be such that generating pre-images satisfying the predicates is easy (if you know the predicate). To generate an x_i, one can pick its last n bits randomly, and define the j-th bit of x_i, for $j = m-n, \ldots, 1$, to be the j-th hidden predicate applied to bits $j+1, \ldots, m$ of x_i. The hidden predicates must be balanced and difficult to learn from random examples. Suggestions of hard-to-learn predicates exist in the learning-theory literature. For example the parity/majority functions of Blum et al.[4] (which are the exclusive-or of some of the input bits together with the majority function on a disjoint set of input bits) are interesting, although slightly more complicated functions may be appropriate in this application when word lengths are short. With $m-n = 32$, the broker can have one hidden predicate for each day of the month. He could reveal a new predicate each day, and ask vendors to check that the coins they receive satisfy these predicates (otherwise the coins will not be accepted by the broker). This would not affect the validity of legitimate coins already in circulation, but makes forgery extremely difficult, since the would-be forger would have to discard much of his precomputation work as each new predicate is revealed. We feel that such techniques are strongly advisable in MicroMint.

Other Extensions

Peter Wayner (private communication) has suggested a variation on MicroMint in which coins of different values are distinguished by publicly-known predicates on the x-values.

5 Relationship to Other Micropayment Schemes

In this section we compare our proposals to the Millicent[10], NetBill [1], NetCard [2], and Pederson [14] micropayment schemes.

NetBill offers a number of advanced features (such as electronic purchase orders and encryption of purchased information), but it is relative expensive: digital signatures are heavily used and the NetBill server is involved in each payment.

Millicent uses hash functions extensively, but the broker must be on-line whenever the user wishes to interact with a new vendor. The user buys vendor-specific scrip from the broker. For applications such as web browsing, where new user-vendor relationships are continually being created, Millicent can place a heavy real-time burden on the broker. Compared to Millicent, both PayWord and MicroMint enable the user to generate vendor-specific "scrip" without any interaction with the broker, and without the overhead required in returning unused vendor-specific scrip. Also, PayWord is a credit rather than debit scheme.

Anderson, Manifavas, and Sutherland [2] have developed a micropayment system, "NetCard," which is very similar to PayWord in that it uses chains of hash values with a digitally signed root. (The way hash chains are created differs in a minor way.) However, in their proposal, it is the bank rather than the user who prepares the chain and signs the root, which adds to the overall burden of the bank. This approach prevents the user from creating new chains, although a NetCard user could spend a single chain many times. Compared to PayWord, NetCard is debit-based, rather than credit-based. We have heard that a patent has been applied for on the NetCard system.

Torben Pedersen outlines a micropayment proposal[14] that is also based on hash chains. His motivating application was for incremental payment of telephone charges. His paper does not provide much detail on many points (e.g. whether the system is credit or debit-based, how to handle exceptions, whether chains are vendor-specific, and other auxiliary security-related matters). The CAFE project has filed for a patent on what we believe is an elaboration of Pedersen's idea. (The details off the CAFE scheme are not available to us.)

Similarly following Pedersen's exposition, the iKP developers Hauser, Steiner, and Waidner have independently adopted a similar approach [8].

6 Conclusions and Discussion

We have presented two new micropayment schemes which are exceptionally economical in terms of the number of public-key operations employed. Furthermore, both schemes are *off-line* from the broker's point of view.

References

[1] The NetBill Electronic Commerce Project, 1995. `http://www.ini.cmu/NETBILL/home.html`.

[2] Ross Anderson, Harry Manifavas, and Chris Sutherland. A practical electronic cash system, 1995. Available from author: `Ross.Anderson@cl.cam.ac.uk`.

[3] Matt Blaze, Whitfield Diffie, Ronald L. Rivest, Bruce Schneier, Tsutomu Shimomura, Eric Thompson, and Michael Wiener. Minimal key lengths for symmetric ciphers to provide adequate commercial security: A report by an ad hoc group of cryptographers and computer scientists, January 1996. Available at `http://www.bsa.org`.

[4] Avrim Blum, Merrick Furst, Michael Kearns, and Richard J. Lipton. Cryptographic primitives based on hard learning problems. In Douglas R. Stinson, editor, *Proc. CRYPTO 93*, pages 278–291. Springer, 1994. Lecture Notes in Computer Science No. 773.

[5] Shimon Even, Oded Goldreich, and Silvio Micali. On-line/off-line digital signatures. In G. Brassard, editor, *Proc. CRYPTO 89*, pages 263–277. Springer-Verlag, 1990. Lecture Notes in Computer Science No. 435.

[6] Phillip Hallam-Baker. W3C payments resources, 1995. `http://www.w3.org/hypertext/WWW/Payments/overview.html`.

[7] Neil M. Haller. The S/KEY one-time password system. In *ISOC*, 1994.

[8] Ralf Hauser, Michael Steiner, and Michael Waidner. Micro-Payments based on iKP, December 17, 1995. Available from authors. `sti@zurich.ibm.com`.

[9] Leslie Lamport. Password authentication with insecure communication. *Communications of the ACM*, 24(11):770–771, November 1981.

[10] Mark S. Manasse. Millicent (electronic microcommerce), 1995. `http://www.research.digital.com/SRC/personal/Mark_Manasse/uncommon/ucom.html`.

[11] Ralph C. Merkle. A certified digital signature. In G. Brassard, editor, *Proc. CRYPTO 89*, pages 218–238. Springer-Verlag, 1990. Lecture Notes in Computer Science No. 435.

[12] Silvio Micali. Efficient certificate revocation. Technical Report TM-542b, MIT Laboratory for Computer Science, March 22, 1996.

[13] National Institute of Standards and Technology (NIST). *FIPS Publication 180: Secure Hash Standard (SHS)*, May 11, 1993.

[14] Torben P. Pedersen. Electronic payments of small amounts. Technical Report DAIMI PB-495, Aarhus University, Computer Science Department, Århus, Denmark, August 1995.

[15] Ronald L. Rivest. The MD5 message-digest algorithm. Internet Request for Comments, April 1992. RFC 1321.

[16] Bruce Schneier. *Applied Cryptography (Second Edition)*. John Wiley & Sons, 1996.

[17] Adi Shamir. Fast signature screening. CRYPTO '95 rump session talk; to appear in RSA Laboratories' *CryptoBytes*.

[18] Peter Wayner. *Digital Cash: Commerce on the Net*. Academic Press, 1996.

http://www.w3.org/pub/www

W3C Technical Reports & Publications

Technical Reports

W3C is chartered to work with its Members to develop technical specifications. Specifications which are given the status of W3C Recommendation, indicating that a general consensus has been reached among Members that the specification is appropriate for use, goes through three stages. The first stage is development of a specification, where the specification is called a Working Draft, or WD. The W3C Director may then promote a working draft to the next stage by issuing it as a Proposed Recommendation, or PR. After the PR has gone through a formal review process by the Members, the W3C Director will determine that the PR:

- will move to the third and final stage, and be issued as a W3C Recommendation; or
- will be issued with minor indicated changes; or
- will be returned for work at the Working Draft stage to address specific issues; or
- will be abandoned and removed from the W3C agenda.

W3C Recommendations

The following Recommendations have been issued:

PNG (Portable Network Graphics) Specification
    1 October 1996, various

W3C Proposed Recommendations

The following Proposed Recommendations are under review by W3C Members:

Rating Services and Rating Systems (and Their Machine Readable Descriptions)
    8 August 1996, Jim Miller, Paul Resnick and David Singer
PICS Label Distribution -- Label Syntax and Communication Protocols
    8 August 1996, Tim Krauskopf, Jim Miller, Paul Resnick and Win Treese
Cascading Style Sheets, level 1
    13 November 1996, Håkon W. Lie and Bert Bos
HTML 3.2 Reference Specification
    5 November 1996, Dave Raggett

W3C Working Drafts

These are working drafts for review by W3C members and other interested parties. These are draft documents and may be updated, replaced or obsoleted by other documents at any time. It is inappropriate to use W3C working drafts as reference material or to cite them as other than "work in progress".

Note: since working drafts are subject to frequent change, you are advised to reference the URL for this page, rather than the URL for working drafts themselves.

Extensible Markup Language (XML)
14 November 1996, Tim Bray and C.M. Sperberg-McQueen
PEP: An Extension Mechanism for HTTP.
19 August 1996, Rohit Khare
HTML3 and Style Sheets.
5 June 1996, Bert Bos, David Raggett and Håkon W Lie
Inserting objects into HTML.
22 April 1996, Dave Raggett, Charlie Kindel, Lou Montulli, Eric Sink, Wayne Gramlich, Jonathan Hirschman, Tim Berners-Lee, Dan Connolly
The HTML3 Table Model.
23 January 1996, Dave Raggett

W3C Notes

The Consortium may make available on the web information, ideas or commentary from W3C staff, members, or the general public. Such information may be released, at the discretion of the W3C Director, as a NOTE.

Imagemapped Images and Image-Incapable User Agents
26 November 1996, Ian Graham
Resource Description Messages (RDM)
24 July 1996, Darren Hardy
Frame-based layout via Style Sheets.
8 June 1996, Bert Bos, David Raggett and Håkon W. Lie
Jigsaw: An Object Oriented Server
7 June 1996, Anselm Baird Smith
Jigsaw performance evaluation
7 June 1996, Anselm Baird Smith
HTML predefined icon-like symbols.
4 June 1996, Bert Bos
A Proposed Convention for Embedding Metadata in HTML.
2 June 1996, Reported by Stuart Weibel
The ILU Requester: Object Services in HTTP Servers
7 March 1996, Paul Everitt
HTML Dialects: Internet Medial and SGML Document Types
6 March 1996, Dan Connolly
Session Identification URI.
21 February 1996, Phillip M. Hallam-Baker, Dan Connolly
Notification for Proxy Caches
21 February 1996, Phillip M. Hallam-Baker
Extended Log File Format.

21 February 1996, Phillip M. Hallam-Baker, Brian Behlendorf
A Lexical Analyzer for HTML and Basic SGML.
8 February 1996, Dan Connolly
Micro Payment Transfer Protocol (MPTP).
22 November 1995, Phillip M. Hallam-Baker
Giving Information About Other Resources in HTML
20 November 1995, Tim Berners-Lee, David Raggett
Proposals for Gathering Consumer Demographics
06 November 1995, Dan Connolly
"Character Set" Considered Harmful
02 May 1995, Dan Connolly

Web Journal

World Wide Web Journal -- the official journal of the W3C, is published quarterly by O'Reilly & Associates.

How to Order W3C Publications

Release Form for contributions to W3C Technical Reports and Specifications

---

Webmaster
$Date: 1996/11/27 18:35:47 $

Micro Payment Transfer Protocol (MPTP) Version 0.1

W3C Working Draft *22-Nov-95*

This version:
http://www.w3.org/pub/WWW/TR/WD-mptp-951122
Latest version:
http://www.w3.org/pub/WWW/TR/WD-mptp ← This document
Authors:
Phillip M. Hallam-Baker <hallam@w3.org>

Status of this document

This is a W3C Working Draft for review by W3C members and other interested parties. It is a draft document and may be updated, replaced or obsoleted by other documents at any time. It is inappropriate to use W3C Working Drafts as reference material or to cite them as other than "work in progress". A list of current W3C working drafts can be found at: http://www.w3.org/pub/WWW/TR

Note: since working drafts are subject to frequent change, you are advised to reference the above URL, rather than the URLs for working drafts themselves.

Abstract

A protocol for transfer of payments through the services of a common broker is described. The processing demands of the protocol make it practical for small payment amounts. The latency makes it practical for use in interactive applications. The scheme thus satisfies the two key criteria for a micropayments scheme. MPTP implements a variation of the Pay-Word proposal of Rivest and Shamir [RivestSh95]. It is also inspired by the Millicent proposal by Manasse [Manasse95] and the iKP proposal by Bellare et. al. [BellareEt95]. A proposal similar to the PayWord scheme by Torben Pedersen [Pedersen9?] was reported after this draft was begun.

For efficiency it is desirable to be able to combine transfer of payments instructions with those accomplishing the delivery of goods. For this reason MPTP may be layered on a variety of Internet protocols including HTTP and SMTP/MIME.

Although the protocol is optimized for use as a payment scheme it is suitable for the transfer of larger amounts. The protocol is also suitable for use as an access control or resource allocation mechanism. With modification the protocol could be made to provide anonymity guarantees.

Contents

- Introduction
    - Principles and Parties
    - Risk Model
    - Policy
    - Mechanism
    - Signature
    - Certificates and establishment of trust.
    - Distributed Implementation
    - Risk Control
        - Credit Liability
        - Credit Abuse
        - Counterfeiting
        - Unauthorized Withdrawal
        - Purchase Order Modification
        - Double Spending
        - Failure to Credit Payment
        - Denial of Service
        - Repudiation
        - Failure to deliver
        - Framing
- Message Formats
    - Account Establishment and Maintenance
        - Create Account
        - Reissue Certificate
        - Delete Account
        - Account Certificate
    - Payment Dataflow
        - Authority
        - Charge
    - Vendor-Broker Communications
        - Collection Request
        - Collection Response
        - Validation Request
        - Validation Response
        - Revocation List
- Processing of Data Flows
    - Account Creation
    - Account Modification, Enquiry, Deletion
    - Payment Flow
    - Collection Flow
- Resource and Performance Analysis
    - Latency
        - Establishment of Payments Session
        - Subsequent Payments
    - Processing
        - Customer

        - Vendor
        - Broker
    - Storage
        - Customer
        - Vendor
        - Broker
- Commentary and Further Work
    - Benchmarks and statistics
    - Risk factors
    - Inter-Broker Settlement Model
    - Wider Applications
- Security Considerations
- Patent Rights
- References.
- Acknowledgements
- Contact
- To Do List

Introduction

Commerce on the Internet may be broadly divided into three categories, advertising, sale of tangible goods and sale of non-tangible goods. Advertising is not directly considered in this paper. Movement of tangible goods requires handling and shipping whose costs set a minimum value for which trading is economic and introduces a substantial delay into the process. Speed and cost of processing are thus not the critical factors in evaluating payment systems for this application. Where non-tangible goods are involved the contract may in many cases be fulfilled through the internet.

There is a large interest in payment systems which support charging relatively small amounts for a unit of information. Here the speed and cost of processing payments are critical factors in assessing a schemes usability. Fast user response is essential if the user is to be encouraged to make a large number of purchases. Processing and storage requirements placed on brokers and vendors must be economic for low value transactions. MPTP is optimized for use for low value transfers between parties who have a relationship over a period of time. It also provides a high degree of protection against fraud making it applicable in wider scenarios, including sale of tangible goods.

The following performance statistics were used as a guide in designing the protocol [RivestSh95]. Current workstation and high end personal computer class machines are capable of approximately two public key signature creation operations, two hundred public key verification operations and 20,000 one way hash functions per second. Latency introduced by round trip communications may be a second or more, the time taken for a signal to be relayed by a satellite in geostationary orbit. Communications also introduce potential unreliability.

A distinction is made between the cost of inline and offline processing. Inline processing takes place on the critical path for payments. Servers must thus have substantial excess capacity to deal with fluctuation in demand. Offline processing may be performed asynchronously when processing load permits. As a rough estimate inline processing was considered to incur two orders of magnitude more cost than offline processing.

In addition to processing costs, the cost of online storage must be borne in mind. If online storage must be accessed during a payment transfer the cost of storing the data in RAM or of disk head contention must be considered.

MPTP is an asynchronous protocol. Much of the processing required may be done offline. In particular payment does not require an online communication with the broker (unless the symmetric signature option is used). MPTP is also symmetric, there is no distinction between customer and vendor accounts except in relation to specific transactions where the flow of payment is generally in a single direction. The ability to make payments need not preclude the ability to accept payments unless this is a matter of broker policy. However in some cases it might be desirable to have different accounts for these functions, a vendor accepting large payments might wish to avoid the danger of a security compromise allowing unauthorized payments to be made via the Internet accept only account.

One of the most significant differences between the World Wide Web and print media is that the cost of publishing on the Web is commensurate with the cost of readership and does not involve a high capital outlay. Many of the initial users of the Web were primarily interested in publishing existing information, both within organizations and to the wider internet community. One important characteristic which a Web micropayment scheme must satisfy is that it permit access to commercial publication to both small and large publishers. MPTP provides for transfer of small payment amounts through vendor amortization. Use of public key signature screening as opposed to verification makes it economic for use by small publishers.

Principles and Parties

MPTP involves three parties, a customer *C* who makes the payment, a vendor *V* who receives the payment and a broker *B* who keeps accounts for the parties concerned. At present only a single broker model is considered, this means that both customer and vendor must share the same broker. Note however that the protocol does not restrict the broker to use of a single server.

Such a capability is essential if customers are to be able to surf the Web using a single payment account and vendors are to be able to accept payments from any source through a single account. Although accounts are in principle being used by both customers and vendors it is likely that brokers will wish to specialize in particular types of account. ISPs for example have an existing client base which is billed on a regular basis. Issuing bills requires a very different type of procedure to issuing payments however and thus vendor support may require the services of a specialist broker. Support for Inter-Broker transfers will be required in the long term to permit the system to be scaled effectively. Some notes on the technical and political difficulties involved are made in the commentary section at the end.

It is important that a payment protocol does not interfere with the established trust relationships between the parties. Where a protocol allows collection of data on another parties activity this should be made clear in advance. It is not a requirement that the protocol duplicate the trust models of a particular financial instrument precisely. It is more important that the protocol provide flexibility in the establishment of trust relationships than attempt to define which party accepts what risk.

The term Broker is used to refer to a financial intermediary. In the context of this proposal a broker might be any organization with the ability to bill a significant number of customers at a small marginal cost. This means that in addition to use by financial institutions MPTP might be suitable for use by Internet Service Providers (ISPs), telephone companies or any other organization sending out large numbers of invoices.

Risk Model

[The risk model will be developed in depth in a companion paper. It is intended that the risk model be comprehensive, permitting cross comparisons between different types of schemes to be made.]

The main risks faced by the customer are liability for unauthorized payment and either not receiving the goods at all or receiving goods different from those advertised. The vendor risks not being paid. The broker risks the cost of customer service due to malfunction or incompatibilities and liability for payments in cases where the customer and vendor are in dispute.

The following risks were considered:

Credit Abuse
  An account is used to make repeated payments without intention to pay.
Counterfeiting
  A fake payment order is constructed.
Unauthorized Withdrawal
  The broker (or an employee) makes an unauthorized withdrawal from an account.
Purchase order modification.
  A customer issues a payment order intending to purchase one set of goods but the order is intercepted and modified by a third party.
Failure to Credit Payment
  The broker debits the customer account but does not credit the vendor account.
Double Spending
  A payment instruction is used twice, either by a customer, a vendor or a third party.
Denial of Service
  A customer or vendor is denied use of their account.
Repudiation
  A party may deny making a payment.
Credit liability
  Where a customer is extended credit liability must be controlled.
Failure to Deliver
  A vendor may accept payment but fail to supply the advertised goods.
Framing
  A party is able to convince another that a third party acted in bad faith.

In many cases it is in the interests of a party to accept risk. A broker may offer to guarantee payments to vendors irrespective of whether the customer pays and require a payment of a higher commission in return. The high cost of customer service enquiries must be borne in mind. Protocols which cannot clearly identify which party was at fault are likely to incur substantial customer service costs.

Policy

The interests of the parties may conflict. In such cases the choice of which parties interest will be prioritized is a matter of policy. MPTP is designed to be policy neutral, permitting a broker to offer a wide number of policy options. Individual vendors may thus choose the precise terms on which they offer goods. Customers may choose the terms they are willing to accept on a vendor by vendor basis

One of the areas in which conflict of interest occurs is whether goods are to be delivered before or after payment. It is in the customers interest to withhold payment until the goods are delivered to ensure that they are satisfactory. The Vendor may wish to ensuring payment is made by insisting of payment in advance however.

As previously noted it may be in the interest of the vendor to encourage customers to purchase goods by accepting a degree of risk. This is particularly important when the vendor has no established reputation with the customer. This risk may be made acceptable to the vendor provided there is a high enough probability that payment will be made. In the green-commerce model [SteinStBoRo94] the broker acts to encourage this by monitoring the number of refusals made by a customer and excluding customers with a bad track record.

MPTP permits a considerable degree of flexibility in establishing a payments policy. A vendor may permit a customer a certain amount of trade before requiring a firm payment commitment or require all purchases to be paid for in advance. The first policy may be applicable where the goods offered cannot be evaluated by the customer in advance. A Vendor who has established a reputation with the customer may be in a position to insist on prior payment.

As an example consider the vendor of a software program who wishes to charge for its use on an hourly basis. The vendor may be willing to offer a customer a free trial provided there is a guarantee that the customer cannot simply request a fresh retrial each time a trial expires.

Mechanism

In the Pay Word scheme a payment order consists of two parts, a digitally signed payment authority and a separate payment token which determines the amount. A chained hash function,is used to authenticate the token. These are described by Lamport [Lamport81] and employed in the S/Key [Haler94] authorization mechanism. To create the payment authority the customer first chooses a value *w_n* at random. The customer then calculates a chain of payment tokens (or *paychain*) *w_0, w_1, ... w_n* by computing

w_i = h (w_i+1)

Where h is a cryptographically secure one way has function such as MD5 [Rivest92c] or SHA [AccreditedSC93a].

The signed payment authority contains *w_0*, the root of the payment chain and defines a value for each link in the chain. Payments are made by revealing successive paychain tokens. Once the vendor or broker has authenticated a payment authority an arbitrary payment token may be authenticated by performing successive hash functions and comparing against the root value. It should be noted however that the broker is only presented with the final payment order. It is therefore unnecessary for the broker to maintain large online databases.

MPTP permits use of double payment chains. This allows implementation of a broker mediated satisfaction guarantee scheme. The pair of payment chains represent the high and low watermarks for the payment order. The low watermark chain represents the amount that the customer has fully committed to pay. The high watermark chain represents partial commitments. The vendor exposure is the difference between the counter values.

MPTP also supports use of multiple payment counters denoting different units of currency. This allows some optimization of processing time through shortening of the payment chains.

MPTP provides protection against double spending through vendor and broker checking of authority identifiers. The size of required Vendor authority identifier matching tables (th *double spending buffer*may be controlled by checking that the authority timestamp is within bounds. An alternative approach would incorporate challenge/response sequence into the session establishment protocol. This could be used to simplify broker double spending prevention measures if constraints were placed on the challenge identifiers. The reduction in vendor resource requirements do not appear to justify an additional round trip delay however.

The mechanism could be modified to use a collection of payment tokens as opposed to a chain. Each token would consist of a the hash of a shared secret which would be revealed to make a payment. This might provide a solution to possible patent difficulties concerning the use of the Lamport hash chain mechanism. It would also permit payments to take place in parallel.

Signature

MPTP permits use of both shared secret and public key based signature schemes. Schneier [Schneier96] describes a wide variety of public key signatures schemes and one way hash functions suitable for constructing Message Authentication Codes (*MACs*). Choice of algorithm, key length etc. is left to the parties involved. It is desirable to minimize the latency introduced in the signing of the initial payment order and also to minimize computational needs of the vendor and broker.

A number of digital signature techniques permit some calculations to be performed offline, i.e. in advance of the message being known [EvenGoMi90]. Such schemes include the Digital Signature Standard [NIST91] and El-Gamal [ElGamal95]. Pre-calculation permits the time taken to generate a signature to be performed outside the human interaction loop and thus appear transparent to the user. Note however that many MPTP applications may chose to perform speculative calculation of an authority in advance of user instructions, unused calculations would simply be discarded. Note however that use of this technique might negatively interact with techniques intended to prevent double spending since a user might delay sending the authority for a significant time.

A recent proposal by Shamir [Shamir95] permits signatures to be validated very rapidly, although at the cost of introducing a small risk that an invalid signature will be accepted. Shamir describes a variation of the Rabin [Rabin79] signature scheme which permits signatures to be "screened" using a test which will detect a fraudulent signature half the time and never reject a valid one. The scheme may be applied repeatedly to provide the desired degree of assurance as to the authenticity of the result.

The signature screening approach has a number of characteristics which are particularly suitable for micropayment schemes. The degree of assurance may be adjusted to reflect the level of risk. Small payments might be accepted with only minimal checking and additional checks performed as the risk increased. another useful property is that the level of protection may be adjusted to reflect server load. This is especially important since it substantially reduces the excess server capacity required to cope with peaks in demand and allows unexpected increases in demand to be dealt with in an orderly manner.

A final signature option is to used a shared secret and keyed digest. This requires the customer and broker to establish a shared secret and for the broker to provide an online verification service. Use of shared secret authentication permits rapid generation of payment authorities. Validation of such authorities requires communication between vendor and broker which may incur a delay.

Certificates and establishment of trust.

Certificates bind a public key to an account number under the public key of the broker. It is assumed that the broker public key is known to all parties. Implementations might require broker public keys to be verified through some additional means.

Each party generates their own public-private key pair locally. The public key certificate is communicated to the broker at account establishment.

The certificate issuance policy may require frequent re-issuance of certificates to enable close control of credit risks or permit certificates to be valid for longer periods of time. It is not necessary for a re-issued certificate to establish a new public key.

Account revocation lists are supported to enable credit risks to be prevented from engaging in further abuse. Separate certificate revocation lists are not supported since compromise of public key certificates may be dealt with through the same mechanism.

Where an account has special attributes concerned with risk management these attributes should be included in and authenticated by the certificate. For example an account might be limited to making payment orders for no more than a certain amount. A broker might chose to guarantee payments up to a certain amount but require an authorization to guarantee payments for larger amounts. Alternatively a broker might require a vendor to accept the risk regardless of amount and authorization. The following certificate attributes are supported:

IP-Address *mask*, *value*

> Specifies a set of internet addresses for which the certificate is valid. Only payment requests originating from IP addresses which equal the specified value after being logically ANDed with the mask. If more than one IP-Address attribute is specified a single match is sufficient.

Not Guaranteed *amount*

> The broker will not guarantee that payment will be made for amounts exceeding the specified amount.

Guaranteed *amount*

> The broker guarantees payment up to the specified amount without separate authorization.

Authorization-Required *amount*

> Payments above the specified amount require separate authorization to be guaranteed.

Distributed Implementation

The offline nature of PayWord lends itself well to a fully distributed solution. It is not necessary for the Broker to use a single server for all customers.

Online verification of credit-worthiness (e.g. in the symmetric signature scheme) requires a vendor to have access to a server which holds the authentication information. This information may be encoded in the account certificate.

Risk Control

Having described the risk model and mechanism of MPTP we describe the manner in which risk may be controlled.

Credit Liability

If a broker chooses to act as guarantor for a payment a credit liability risk may be incurred. Note that MPTP supports an option for the broker to transfer this risk to the vendor by refusing a guarantee of payment.

In either case a credit liability is incurred. Such liabilities are a familiar consideration in the financial industry. A similar risk is accepted in parts of the publishing industry where newspapers are sold from unattended vending machines which cannot control the number of copies taken by each customer. In certain countries no precautions are taken to prevent a copy being taken without any payment at all.

Credit Abuse

The problem of credit abuse is linked to but distinct from that of credit liability risks. For example an account might be created in a false name and its authentication information widely published with the intention of permitting general access to charged material for free.

Credit abuse might be discovered through broker tracing of payment patterns to detect sudden increases in payment activity and then terminated through the revocation list mechanism. The case of widespread use of a single connection may be controlled through checking of the certificate IP-Address attribute if specified. If no IP address attribute is specified a vendor might employ code to detect accesses from multiple IP addresses within a suspiciously short interval.

Counterfeiting

MPTP payment orders are vendor specific and digitally signed. Provided the signature scheme is secure it is not possible for a party to construct a payment order without having access to the secret information corresponding to the key.

Unauthorized Withdrawal

Unauthorized withdrawal is not possible without detection by the account holder who may require an audit trail from the broker for each transaction. Note that this requires the broker to maintain a substantial quantity of online logging information.

Purchase Order Modification

In a purchase order modification attack an external party modifies a purchase request in order to cause different goods to be delivered. This risk is not directly addressed in the MPTP scheme although the satisfaction guaranteed policy might be used to protect the customer.

Without authentication of the purchase order there is no method of avoiding this attack. The cost of this authentication might be reduced by establishing a shared key between vendor and customer during the session establishment protocol. Such shared keys might have a lifetime spanning several payment orders.

Double Spending

Payment orders are specific to a particular vendor and carry a unique authority identifier. A broker is required to detect an attempt to deposit the same payment order more than once and act accordingly. In some cases this may mean increasing the amount of payment authorized.

Failure to Credit Payment

Currently MPTP does not address this risk. A Broker may deliberately deduct a payment amount from the account of one party without making a corresponding credit to another party.

One approach to this problem is to make information concerning bad debts available for scrutiny. A broker might be required to issue a frequent list of bad debts signed under the broker's public key. Such debts might be rendered unlinkable through a use of a one way hash function on the authority identifier. The proportion of bad debts might be concealed through addition of padding. In this way both customer and vendor could ensure that the broker acted in good faith.

Denial of Service

Denial of service is a significant risk, unfortunately it is one that the underlying infrastructure of the Internet does not protect against. Consequently any application protocol level protection against a denial of service attack can at best provide limited protection against this risk.

Use of Shamir's signature screening algorithm substantially reduces the risk of a denial of service attack against a vendor or broker through construction of bogus payment orders.

Repudiation

MPTP payment orders are non-repudiable in the sense that the customer cannot deny having made a payment authorization. This is distinct from the option for a vendor or broker to permit a customer the right to refuse payment after receiving the goods.

Failure to deliver

Failure to deliver may occur for many reasons including vendor fraud. The Internet is an unreliable transport medium and a customer may in good faith offer to buy an article and a vendor in good faith may intend to supply but delivery fail nevertheless. The HTTP protocol in particular does not currently provide for customer acknowledgment of receipt.

One solution to the failure to deliver risk is to permit the customer to refuse payment through the "satisfaction guaranteed" policy described earlier.

Framing

The vendor has the opportunity to frame a customer, albeit at a direct monetary loss to himself. In this scenario a vendor receives a valid payment chain from a customer but chooses not to deliver the authorization paychain token, instead delivering only the promissory paychain token. The vendor is thus able to frame the customer, albeit at the cost of the payment.

This risk is not currently addressed in MPTP. One approach to addressing this risk would be for the customer to opt to make the payment during account reconciliation.

Message Formats

In this draft we describe the message content without entering into consideration of the corresponding byte streams. We assume that a systematic encoding of these message formats is employed such as the Basic Encoding Rules ASN.1. Choice of encoding rules is left to the working group. It is assumed that the encoding permits payment messages to be transport via standard Internet protocols through simple processing (e.g. BASE-64 encoding).

We define the following data types which are of general use:

```
Identifier : Array [Octet]

Amount : Struct
    value                    Integer
    currency                 Identifier

Signed [Any] : Struct
    data                     Any
    certificate              AccountCertificate
    signature_algorithm      Identifier
    signature                Identifier

Wrapper [Any] : Struct
    version                  Identifier
    message-id             Identifier
    data                     Any
```

All messages are transported enveloped using the `Wrapper` structure which states the protocol version and message identifier.

Account Establishment and Maintenance

The following account options are supported:

Accept Payments Only
    The account will only accept payment.
Refuse Payments
    The account does not accept payments of any type.
Refuse Micropayment
    The account permits payments to be made to it but cannot initiate micropayments. Such accounts may be useful for customers who may wish to be able to transfer money into their account but do

not require the ability to accept large numbers of arbitrarily small payments.

Create Account

Account creation requires a binding to be established between an account identifier supplied by the broker, a public key supplied by the potential client and billing information. The exact nature of the billing information is left to implementations.

```
CreateAccountRequest : Struct
    public_key                PublicKey
    identity_binding          ???

AccountFlag : Choice
    AcceptPaymentsOnly
    RefusePayments
    RefuseMicroPayments
```

It might be appropriate to encrypt the identity binding information.

Reissue Certificate

Depending on broker policy certificates may require frequent reissue. This process may or may not require the establishment of a new public key. Note that this is not a suitable mechanism for dealing with certificate compromise situations.

```
ReissueCertificateRequest : Signed [ReissueCertificateData]

ReissueCertificateData : Struct
    account_id                Identifier
    public_key                PublicKey
```

Delete Account

The delete account message is used to terminate an account.

```
DeleteAccountRequest : Signed [DeleteAccount]

DeleteAccount : Struct
    account_id           Identifier
```

Account Certificate

The account certificate is returned by the broker in response to an account creation request.

```
AccountCertificate : Signed [AccountData]

AccountData : Struct
    account_id                Identifier
    flags                     Set[AccountFlags]
    credit_limit              Amount
    broker_servers            List [BrokerServer]
    attributes                List [Attribute]
    not_valid_before          Date
    not_valid_after           Date
```

```
Attribute : Choice
    IPAddress
        mask                        Address
        value                       Address
    NotGuaranteed
        amount                      Amount
    Guaranteed
        amount                      Amount
    AuthorizationRequired
        amount                      Amount
```

Payment Dataflow

The payment dataflow consists of three phases. First an account authority is created, next a sequence of paywords is transferred, finally a termination message closes the payment session. Payment sessions may span multiple transport sessions.

A payment authority may optionally incorporate a direct payment instruction which does not require confirmation using a pay-chain.

Authority

```
Authority : Signed [AuthorityData]

AuthorityData : Struct
    version                         Identifier
    authority_id                    Identifier
    payer_id                        Identifier
    recipient_id                    Identifier
    date                            Date
    hash_algorithm                  Identifier
    chains                          List [PayChain]

PayChain : Struct
    flags                           Set[ChainFlag]
    amount                          Amount

ChainFlag : Choice
    Accepted
    Pending

BrokerServer : Struct
    address                         Array [Octet]
    port                            Array [Octet]
    quality                         Integer
```

It is assumed for the sake of convenience that all pay-chains under a given authority will employ the same hash function.

Charge

```
Charge : Struct
    authority_id                    Identifier
    paywords                        List [PayWords]
    terminate                       Boolean

PayWord : Struct
```

```
    pay_word                  Array [Octet]
    increment                 Integer
```

The terminate flag terminates a payments session and informs the vendor that no further payments are to be made.

Vendor-Broker Communications

The collection process permits the vendor to collect payment on paychains

Collection Request

A collection request consists simply of a list of authority, charge pairs.

```
CollectionRequest : Struct
    account_id                Identifier
    collections               List [Collection]

Collection : Struct
    authority                 Authority
    charge                    Charge
```

Collection Response

There is no need to respond with an affirmation for every payment. Simply provide a list of the duds.

```
CollectionResponse : Struct
    status                    BrokerStatus
    refusals                  List [ChainResponse]

ChainResponse : Struct
    authority_id              Identifier
    reason                    RefusalReason
```

Validation Request

Validation requests are required whenever symmetric key signatures are employed. Validation requests might also be required as a matter of broker policy in certain circumstances, such as purchases for large amounts or to guarantee payments.

```
ValidationRequest : Struct
    vendor_id                 Identifier
    authority                 Authority
```

Validation Response

```
ValidationResponse : Struct
    broker_id                   Identifier
    vendor_id                   Identifier
    authority_id                Identifier
    response                    ValidationResponseCode
ValidationResponseCode : Choice
    Authorized
    NotAuthorized
```

Revocation List

```
RevocationList : List [Revocation]

Revocation : Struct
    account_id                  Identifier
    reason                      RevocationReason
```

Processing of Data Flows

[This is a placeholder for detailed descriptions of the required processing steps.]

Account Creation

[To Be Specified]

Account Modification, Enquiry, Deletion

[To Be Specified]

Payment Flow

Session Establishment [Customer]

The customer performs the following steps to create an `Authority`:

1. Calculates PayChains, stores head, may additionally store all or part of PayChain.
2. Creates unique authority identifier. Alternatively the paychain root might be used for this purpose.
3. Fills remaining slots in Authority structure.
4. The authority is sent to the vendor.

Session Establishment [Vendor]

On receipt of an `Authority` the vendor performs the following steps:

1. The date of the authority is checked to ensure that it is within the vendor determined permitted timeframe.
2. The authority identifier is checked against those in the double spending check buffer.
3. The authority identifier is added to the double spending check buffer. [The vendor may opt to remove expired entries from the double spending check buffer at this time].
4. If public key signatures are used
   the signature of the customer certificate validated.
5. If public key signatures are used:
   The signature of the authority is validated.
6. If the account certificate does not offer the required payment guarantees or symmetric signatures are used:
   A validation request is performed.
7. The Authority is appended to the online file.

Session Establishment with Validation Request [Vendor]

If the vendor determines that an account enquiry is required an account enquiry is created:

1. The account enquiry packet is created
2. The account enquiry is authenticated using a MAC and a shared secret established between

vendor and broker.
    3. The account enquiry is sent to the broker.

Session Establishment with Validation Request [Broker]

1. A Validation Request is received.
2. The validation request is authenticated
3. The account information corresponding to the customer id is retrieved.
4. A decision is made to accept or reject the authorization.
5. The Validation Response is sent to the Vendor

Session Establishment with Validation Request [Vendor]

On reciept of an account enquiry response a vendor:

1. Checks to see that the response is genuine.
2. Checks to see that the account is authenticated and the required payments guarantees provided.

The Customer may then send a sequence of Paywords which are processed as follows:

Payment Transfer [Customer]

The Customer prepares a Charge message as follows:

1. The Authority information corresponding to the vendor id is retrieved.
2. The Payword(s) corresponding to the desired payment amount is determined.
3. The Charge message is sent to the vendor.

Payment Transfer [Vendor]

The Vendor processes the Charge message as follows:

1. The vendor receives the charge message.
2. The session record is retrieved using the authority-id.
3. The payword is validated using the paychain root.
4. The payword information and increment are updated in the session record.

Collection Flow

[To Be Specified]

Resource and Performance Analysis

The critical features distinguishing a micropayments protocol from a payments protocol are low latency, low processing requirements and low storage requirements.

Latency

Processing of MPTP micropayments should not introduce noticable delay into the user interaction of a well written application program during normal browsing patterns.

Establishment of Payments Session

Establishment of a payment session requires one digital signature to be generated and two signatures to be checked plus the generation of one or more paychains. Paychain generation and part of the signature generation may be performed offline as a background task, reducing the latency of the interation.

Note that in many cases a sophisticated customer application program may perform the entire process of creating an authority on a speculative basis before the user requests a session to be established. This carries no risk since an authority does not allow payment unless accompanied by a valid payword and in any case the authority would not be sent to the vendor unless a session was to be established.

Subsequent Payments

Subsequent payments require only the generation of the next payment token in the chain and its verification. The generation process may be accelerated or avoided entirely through partial or complete caching of the original paychain.

Processing

The most common processing operation are those connected with the payments dataflow itself.

Customer

The customer bears the most substantial processing costs. Establishment requires the creation of a paychain and digital signature. Offline signature techniques and pre-calculation of pay-word chains may often be performed as background tasks while the processor is idle.

Vendor

The vendor must process two signature verifications per establishment of a payment session and one hash operation per payword transferred, two if double chains are employed.

Hash chain and signature calculations would normally be calculated inline. Use of signature screening might be combined with signature verification to control the inline/offline calculation load.

Broker

The broker must perform one signature verification per collection, plus one hash calculation per payword transferred. It may be possible for a broker to perform probabilistic checking of collection operations, checking only ten percent of a vendors collection request.

All broker calculations may be performed offline.

Storage

Many proposed micropayment schemes offer low processing overhead but require large quantities of data to be kept online for rapid access. Where the frequency of incomming requests is high online access cannot be satisfactorily provided by secondary storage such as disks since head contention becomes the limiting factor. Online storage requirements are thus effectively RAM storage requirements.

Offline storage requirements are unlikely to be a significant factor in the economics of a payments scheme. Many existing servers handle a heavy load of incoming requests while keeping comprehensive log files.

Customer

The customer must track each open session. It may in addition be desirable to store the computed paychains in complete or partial form.

Vendor

The vendor must maintain an online record for each open session. This record is fixed in length consisting of the authority identifier and payer identifier from the authority, and the paychain root or most recent valid pay-word plus the currency unit.

Prevention of double spending requires the maintenance by the vendor of an online record of all payment sessions established within the timestamp validity window. Note however that it may be desirable to place loose limits on validity windows to permit use of speculative calculation of authorities.

A server satisfying 100,000 micropayment operations per hour of which 10% are session establishment requests would require only 8Mb of online storage for both recording of current sessions and maintenance of the double spending prevention window. Such a server would generate $1000 per hour at a cent per transaction which would be more than enough at present prices to meet the cost of the memory.

Broker

If the symmetric signature option is not provided the broker may perform almost all operations offline in batch. Incoming collection requests from vendors may be pre-processed to optimize access to secondary storage such as disk. Detection of double spending requires a record of all transactions to be available at the time when a record is added. This need not involve the expense of online memory however.

One way round the double spending problem is to give each vendor a counter which must be incremented at each step. It is then only necessary to keep one online storage location per account. Note however that it is undesirable that this token be advertised to the customer since it would reveal the number of purchase requests made to that vendor. Another problem would be enforcing the serialization of the tokens, what would happen if one customer terminated a session much later than one started after it? This would seem to imply that the serialization option would require rapid redemption of the tokens which is itself undesirable.

If the symmetric signature option is provided the registry of shared secrets must be available in primary storage. In most practical schemes this will require the data to be stored in RAM.

Commentary and Further Work

This proposal is considered incomplete and comments are invited. A number of additional considerations which might be explored are noted below.

Benchmarks and statistics

It would be useful to have timings for the various processes involved and more comprehensive estimates of relative costs. Detailed statistics concerning customer browsing patterns would be an advantage. How frequently does a customer change site, what proportion of one off purchases does a customer make?

Risk factors

Statements concerning the relative importance of various risk factors to potential customers, vendors and brokers would be of assistance.

Inter-Broker Settlement Model

The protocol described requires perfect trust between servers acting as brokers. Compromise of one server compromises all others. The provisions allowing for multiple servers are not designed to permit multiple brokers competing brokers to participate within a single payment system.

In a large scale use more than one organization would offer broker services but payment transfer should be possible nevertheless even if the parties did not have a common broker. Each transaction may thus involve two brokers, in credit card terminology an acquiring (vendor appointed) bank and issuing (customer appointed) bank. In addition the services of a clearing house may also be required. Such a provision is probably essential to the long term acceptability of a scheme.

The chief difficulty in extending the scheme concerns the establishing to what trust relationships may be assumed between brokers and to what degree enforcement mechanisms must be provided for. If brokers are able to establish a high degree of trust the impact upon the protocol is small. If brokers are unable to establish such trust the impact might be large.

As an example of a naïve Inter-Broker settlement scheme let us consider extending the certification hierarchy to include one or more broker certification authorities. We assume that inter-broker settlement takes place either through direct exchange of payment orders or employs the services of a clearing house. The only direct impact of these changes as far as the customer and vendor are concerned is the need to authenticate the broker certificate.

The impact on the broker trust relationship is more complex. In particular a customer's broker has the opportunity to commit fraud with negligible probability of detection. On receipt of a payment instruction a customer broker might deduct the amount from the customer's account but report it as bad credit to the vendor's broker.

A more subtle problem concerns the trust relationships between the vendor and the vendor's broker. In the credit card system this relationship is transparent. The identity of the vendor's broker is not revealed either to the customer or even the customer's broker. While this is a significant disadvantage for a scheme intended to use the credit card charging infrastructure as previously noted this need not be the case.

Many of the trust and information sharing issues connected with the introduction of inter-broker settlement may be rendered moot by the nature of the initial acceptance community. In particular the question of which party bears what risk is central to this issue. It is therefore premature to make a detailed proposal concerning this issue.

Wider Application

MPTP is a general resource management protocol. It might also be used for control of resources such as printer pages [Hallam-Baker 95b], CPU time and similar low unit cost items within an institution. MPTP might also be applied to provide resource constraint enforcement in applications such as interactive multi-player games (e.g. MUDs, MOOs). [Hallam-Baker95a]

Another potential application area is authorization. MPTP might be used to establish generalised and decentralized authorization in a distributed environment in a similar manner to Kerberos.

Security Considerations

This whole document is about security

Patent Rights

A number of companies sell patent rights to public key technology. The legal status of a number of these patents is currently disputed. Until then the standard IETF spiel with a revision to the PKP bit will appear here. Reports have also surfaced that the CAFE consortium may have a patent covering certain uses of S/Key technology with respect to payments applications.

References.

[AccreditedSC93a] *Working Draft: American National Standard X9.30-1993: Public Key Cryptography Using Irreversible Algorithms for the Financial Services Industry: Part 2: The Secure Hash Algorithm (SHA)* Accredited Standards Committee X9 1993,

[BellareEt95]
Internet Keyed Payment Protocols (iKP) Mihir Bellare, Juan A. Garay, Ralf Hauser, Amir Herzberg, Hugo Krawczyk, Michael Steiner, Gene Tsudik, Michael Waidner: In Proceedings of the First USENIX Workshop on Electronic Commerce, New York, July 1995

[ElGamal85]
A Public Key Cryptosystem and a Signature Scheme Based on Discrete Logarithms. T. El Gamal. IEEE Trans. Inform. Theory, Vol.31, 1985, pages 469--472

[EvenGoMi90]
Online/offline digital signatures. S. Even, O. Goldreich and Silvio Micali. CRYPTO 89, pages 263-277. Springer-Verlag, 1990, Lecture Notes in Computer Science No. 435.

[Haler94]
The S/Key One-Time password system.. N. M. Haller, In ISOC 1994

[Hallam-Baker95]
Electronic Payment Schemes. P. M. Hallam-Baker.
`http://www.w3.org/pub/WWW/Payments/roadmap.html`

[Hallam-Baker95a]
Multi User Dungeons P. M. Hallam-Baker

[Hallam-Baker95b]
Why Printing Should be via The Web P. M. Hallam-Baker

[Lamport81]

Password Authentication with Insecure Communication L. Lamport, Communications of the ACM Nov 1981 pages 770-771.

[Manasse95]

Millicent (electronic microcommerce) Mark S. Manasse. 1995.

[NIST91]

Digital Signature Standard (DSS) National Institute for Standards and Technology Federal Register, vol 56 No. 169 August 1991.

[Pedersen9?]Technical report? Torben Pedersen, DAIMI, Aarhus

[Rabin79]

Digital Signatures and Public Key Functions as Intractable as Factorization". M. O. Rabin. MIT Laboratory for Computer Science Technocal Report, MIT/LCS/TR-212 Jan 1979.

[Rivest92c]

RFC 1321, The {MD5} Message-Digest Algorithm R. L. Rivest, Internet Request for Comments, April 1992

[RivestSh95]

PayWord and MicroMint--Two Simple Micropayment Schemes, R.L. Rivest and A. Shamir. (To Appear.)

[RivestShAd78]

A Method for Obtaining Digital Signatures and Public-Key Cryptosystems R. L. Rivest, A. Shamir and L. M. Adleman. CACM, Feb 1978 p 120-126.

[Schneier96]

Applied Cryptography (Second Edition). Bruce Schneier, John Wiley & Sons 1996.

[Shamir95]

Fast Signature screening. CRYPTO'95 rump session talk. to appear in RSA Laboratories *Cryptobytes*.

[SteinStBoRo94]

The Green Commerce Model L. H. Stein, E. A. Stefferud, N. S. Borenstein and M. T. Rose, Internet Draft. October, 1994, Work in Progress.

Acknowledgements

Thanks are due to Ron Rivest and Adi Shamir for their suggestion to use a chained hash function. The help and advice of Rohit Khare, Dave Ragget, Jim Gettys and Mark Manasse were of great assistance in producing this proposal.

Contact

The author may be contacted via email at hallam@w3.org

To Do List

- Check over symmetric key mode
- Firm up language, inline/offline poorly explained, establishment of payment session etc. Label each term introduced and check for definition. [would not a tool for this be nice]
- Message id business is poorly explained.
- Should firm up the collection loop semantics. It is not absolutely essential that payments be collected only once. If the server can check against double spending could allow partial and

incremental collection

- Consider leakage of data to various parties.
- Protection against double spending by customer currently requires each vendor to maintain an online check of all previous payments by the customer. This is weak protection and could be firmed up substantially.
- Include a challenge response loop to initiate the establishment instruction, thus ensuring that double spending cannot take place?
- Response messages by the broker should be considered somewhat.
- Consider the specific case of payment for software on an hourly basis in detail.
- The issue of maintenance of blacklists by individual merchants requires special attention.
- Should there be a reputation mechanism built into the system so that poor payers suffer a declining credit rating which is advertised in their certificate?
- Need to consider the issue of account enquiries. There should be a mechanism whereby a client can rapidly ascertain that the broker account is correct, establish which funds are cleared etc.
- Should there be a validity interval for payments built into each payment order (cash by date) built into each authority?
- Should there be a do not pay before date in a payment authority?
- Additional references: Kerberos, HTTP, SMTP, MIME.

http://robotics.stanford.edu/users/ketchpel/ecash.html
(9 Oct 96)

A Bibliography of Electronic Payment Information

Digital Cash

1. Cross Industry Work Team Report
2. David Chaum's DigiCash
3. ISI's NetCash
4. Stefan Brands digital cash
5. CyberCash, Inc.
6. NetBank's NetCash
7. Magic Money
8. Sandia Lab's Electronic Cash

Anonymous Credit Cards

1. Bell Labs Description
2. Analysis of Collusion in above

Putting it all together

1. Using Anonymous Credit Cards to pay for services
2. IISP's call for standards for Payment & Billing
3. iKP Electronic Payment Protocols from IBM
4. Bell Labs WWW-buyinfo Mailing List & Archive
5. CyberCash Credit Card Protocol Version 0.8
6. Netscape's Credit card payment applications over the internet
7. MIT Simple Network Payment Protocol
8. CMU's NetBill System
9. Building on NetBill for privacy
10. Open Market System
11. First Virtual
12. PayMe: Secure Payment for World Wide Web Services
13. CARI (Collect All Relevant Information)
14. Clickshare
15. GC Tech's Globe ID
16. VISA/MasterCard Secure Electronic Transaction (SET)
17. Subscrip
18. Ziplock
19. Vishnu from HP Labs in Bristol, A Secure, Cheap and Exportable/Importable Method for Internet Electronic Payments
20. CAFE
21. NetPay
22. Inter-Mark, credit card processing software

23. WebCharge
24. Digital Money, collect payments via modem, phone, fax, mail
25. Interval
26. Electronic Funds Clearinghouse

Electronic Checks

1. BankNet
2. Checkfree
3. Electronic Funds Clearinghouse, Inc.
4. FSTC Electronic Check Project
5. NetCheque
6. NetChex
7. Online Check Systems
8. Redi-Check, pre-authorized check drafts
9. Intell-a-Check

Reports

1. E-Money FAQ compiled by Jim Miller
2. Internet Banking White Paper from WebTech
3. Agenda for IETF project
4. An Application Level Internet Payment Syntax
5. Electronic Money and Money in History
6. Journal of Internet Banking and Commerce
7. Notational Money in Electronic Commerce
8. Clifford Neuman's Payment Tutorial

Micropayments

1. Micro Payment Transfer Protocol (MPTP)
2. Millicent
3. PayWord and MicroMint

Smart Cards

1. Mondex on the Net
2. NetFare

Pointers to Pages of Pointers

1. Matthew Pierce's List of Electronic Payment Systems
2. SIRENE's collection on Electronic Commerce
3. W3C's list of Payment Systems
4. GNN's list of Network Money Sites on the Internet

5. Cornell List
6. WWW-buyinfo hosted at Bell Labs
7. Yahoo's list of Digital Money
8. e$ sites of interest, compiled by Robert Hettinga
9. EFF's collection on Digital Money
10. NetBill's collection of related pointers
11. EIT's list of e-commerce projects
12. CommerceNet's Electronic Commerce Jump Station
13. Hal Varian's Electronic Commerce Links
14. Brad Cox's list
15. Jeff MacKie-Mason's extensive list of Net Commerce resources

```html
<HTML>

<HEAD>
<TITLE>
A Bibliography of Electronic Payment Information
</TITLE>
</HEAD>
<H1>
A Bibliography of Electronic Payment Information
</H1>
<H2>
Digital Cash
</H2>
<OL>
<LI><A HREF="http://www.cnri.reston.va.us:3000/XIWT/documents/dig_cash_doc/ToC.html">Cross
Industry Work Team Report</A>
<LI> <A HREF="http://www.digicash.com/">David Chaum's DigiCash</A>
<LI><A HREF="http://www.netcheque.org/info/netcash">ISI's NetCash</A>

<LI> <A HREF="http://www.cwi.nl/~brands/cash.html">Stefan Brands digital cash</A>
<LI> <A HREF="http://www.cybercash.com/">CyberCash, Inc.</A>
<LI> <A HREF="http://www.teleport.com/~netcash/">NetBank's NetCash</A>
<LI> <a href="ftp://ftp.u.washington.edu/public/phantom/cpunk/mgmny.html">Magic Money </a>
<LI><A HREF="http://www.cs.sandia.gov/HPCCIT/el_cash.html">Sandia Lab's Electronic
Cash</A>
</OL>
<H2>
Anonymous Credit Cards
</H2>
<OL>
<LI> <A HREF="ftp://ftp.research.att.com/dist/anoncc/anoncc.ps.Z">Bell Labs
Description</A>
<LI> <A HREF="ftp://ftp.research.att.com/dist/anoncc/collude.ps.Z">Analysis of Collusion
in above</A>

</OL>
<H2>Putting it all together</H2>
<OL><LI> <A HREF="ftp://ftp.research.att.com/dist/anoncc/accinet.ps.Z">Using Anonymous
Credit Cards to pay for services</A>
<LI> <A HREF="http://www.ansi.org/iisp/need34.html">IISP's call for standards for Payment
& Billing</A>
<LI> <A HREF="http://www.zurich.ibm.ch/Technology/Security/extern/ecommerce/iKP.html">iKP
Electronic Payment Protocols from IBM</A>
<LI> <A HREF="http://www.research.att.com/www-buyinfo/index.html">Bell Labs WWW-buyinfo
Mailing List & Archive</A>
<LI> <a href="ftp://ftp.cybercash.com/pub/draft-cybercash-v08-00.txt">CyberCash Credit
Card Protocol Version 0.8</a>
<li><a href="http://home.netscape.com/newsref/std/credit.html">Netscape's Credit card
payment applications over the internet</a>
<LI> <A HREF="ftp://ana.lcs.mit.edu/pub/snpp/snpp-paper.ps.Z">MIT Simple Network Payment
Protocol</A>
<LI> <A HREF="http://www.ini.cmu.edu/netbill/">CMU's NetBill System</A>
<LI> <A HREF="http://www.ini.cmu.edu/~thoth/ms-thesis-abstract.html">Building on NetBill
for privacy</A>

<LI> <A HREF="http://www.openmarket.com/">Open Market System</A>
<LI> <A HREF="http://www.fv.com/">First Virtual</A>
<LI> <A HREF="http://www.w3.org/pub/Conferences/WWW4/Papers/228/">PayMe:  Secure Payment
for World Wide Web Services</A>
<LI><A HREF="http://www.netresource.com/itp/cari.html">CARI (Collect All Relevant
Information)</A>
<LI><A HREF="http://www.clickshare.com/clickshare/">Clickshare</A>
<LI><A HREF="http://globeid.gctech.fr/cgi-gid/welcome?Language=English">GC Tech's Globe
ID</A>
<LI><A HREF="http://www.visa.com/cgi-bin/vee/sf/set/intro.html">VISA/MasterCard Secure
Electronic Transaction (SET)</A>
```

```
<LI><A HREF="http://www.cs.newcastle.edu.au/Research/afurche/subscrip.ps">Subscrip</A>
<LI><A HREF="http://www.portsoft.com/">Ziplock</A>
<li><a href="http://www.hpl.hp.co.uk/projects/vishnu/main.html">Vishnu from HP Labs in
Bristol, A Secure, Cheap and Exportable/Importable Method for Internet Electronic Payments
</a>
<li><A HREF="http://www.informatik.uni-hildesheim.de/~sirene/projects/cafe/">CAFE</a>
<LI><a href="http://www.bosauto.com/">NetPay</a>

<li><A HREF="http://www.tech-comm.com/">Inter-Mark, credit card processing software</a>
<LI><A HREF="http://www.webcharge.com/">WebCharge</a>
<li><a href="http://www.digimoney.com/">Digital Money, collect payments via modem, phone,
fax, mail</A>
<LI><A HREF="http://www.interval.net/">Interval</a>
<LI><A HREF="http://www.efunds.com">Electronic Funds Clearinghouse</A>
</OL>

<H2>Electronic Checks</H2>
<OL>
<li> <A HREF="http://mkn.co.uk/bank">BankNet</A>
<LI> <A HREF="http://www.checkfree.com/">Checkfree</A>
<LI><A HREF="http://www.efunds.com/">Electronic Funds Clearinghouse, Inc.</A>
<LI><A HREF="http://www.llnl.gov/fstc/index.html">FSTC Electronic Check Project</A>
<LI><A HREF="http://www.netcheque.org/info/netcheque">NetCheque</A>
<LI><A HREF="http://www.netchex.com">NetChex</A>
<LI><A HREF="http://www.onlinecheck.com/">Online Check Systems</A>
<LI><A HREF="http://www.redi-check.com/">Redi-Check, pre-authorized check drafts</A>
<li><a href="http://www.icheck.com/">Intell-a-Check</a>
</OL>

<H2>Reports</h2>
<OL>
<LI><A HREF="http://www.ex.ac.uk/~RDavies/arian/emoneyfaq.html">E-Money FAQ compiled by
Jim Miller</A>
<LI><A HREF="http://www.sfnb.com/wpaper.html">Internet Banking White Paper from
WebTech</A>
<LI><A HREF="http://www.imc.org/ietf-pay/ietf-pay-charter">Agenda for IETF project</A>
<LI><a href="http://www.research.att.com/www-buyinfo/archive/1173.html">An Application
Level Internet Payment Syntax</a>
<LI><a href="http://www.ex.ac.uk/~RDavies/arian/money.html">Electronic Money and Money in
History</a>
<li><A HREF="http://www.arraydev.com:80/commerce/JIBC/">Journal of Internet Banking and
Commerce</A>
<li><A href="http://www.cs.cmu.edu/afs/cs.cmu.edu/user/jeanc/www/usenix.html">Notational
Money in Electronic Commerce</a>
<LI><A HREF="http://nii.isi.edu/people/bcn/talks/network-payment-tutorial">Clifford
Neuman's Payment Tutorial</A>

</ol>

<h2>Micropayments</h2>
<ol>
<li><A HREF="http://www.w3.org/hypertext/WWW/TR/WD-mptp">Micro Payment Transfer Protocol
(MPTP)</A>
<li><A HREF="http://www.research.digital.com/SRC/millicent/">Millicent</A>
<LI><A HREF="http://theory.lcs.mit.edu/~rivest/RivestShamir-mpay.ps">PayWord and
MicroMint</A>
</ol>

<h2>Smart Cards</h2>
<ol>
<li><A HREF="http://www.mondex.com/mondex/net.htm">Mondex on the Net</A>
<li><A HREF="http://www.netfare.com/">NetFare</A>
</ol>

<H2>Pointers to Pages of Pointers</H2>
<OL>
```

```
<LI>Matthew Pierce's <A HREF="http://ganges.cs.tcd.ie/mepeirce/Project/oninternet.html">Li
st of Electronic Payment Systems</A>
<LI>SIRENE's collection on <A HREF="http://www.zurich.ibm.com/Technology/Security/sirene/o
utsideworld/ecommerce.html#syst">Electronic Commerce</A>
<LI>W3C's list of <A HREF="http://www.w3.org/hypertext/WWW/Payments/roadmap.html">Payment
Systems</A>
<LI> <a href="http://gnn.com/meta/finance/feat/links.html">GNN's list of Network Money
Sites on the Internet</a>
<LI> <a href="http://abh.cit.cornell.edu/ecom.html">Cornell List</a>
<LI><a href="http://www.research.att.com/www-buyinfo/index.html">WWW-buyinfo hosted at
Bell Labs</a>
<LI><a href="http://www.yahoo.com/Business_and_Economy/Electronic_Commerce/Digital_Money/"
>Yahoo's list of Digital Money</a>
<LI><a href="http://thumper.vmeng.com/pub/rah/sites.html">e$ sites of interest, compiled
by Robert Hettinga</a>
<LI> <A HREF="http://www.eff.org/pub/Privacy/Digital_Money">EFF's collection on Digital
Money</A>
<LI><A HREF="http://www.ini.cmu.edu/NETBILL/commerce.html">NetBill's collection of related
pointers</A>
<LI><A HREF="http://www.eit.com/~ali/INDEX/ecproj.html">EIT's list of e-commerce
projects</A>
<LI><A HREF="http://www-1.openmarket.com:80/personal/grant/ecjs/payment.htm">CommerceNet's
Electronic Commerce Jump Station</A>
<LI><A HREF="http://www.sims.berkeley.edu/resources/infoecon/Commerce.html">
Hal Varian's Electronic Commerce Links</A>
<LI><A HREF="http://www.virtualschool.edu/mon/ElectronicMoney.html">Brad Cox's list</a>
<LI><A HREF="http://www.spp.umich.edu/telecom/net-commerce.html">Jeff MacKie-Mason's
extensive list of Net Commerce resources</A>
</OL>
</BODY>
</HTML>
```

~~www.cosmo21~~ www.cosmo21.com/netcash/index.htm

NetCASH COSMO21

Electronic commerce, Electronic Data Interchange (EDI), and digital cash are some of the leading technologies

that are rapidly changing the way we do business today. In the NetCASH section here, you will find pointers

to ELECTRONIC COMMERCE RESOURCES and articles that keep you up-to-date.

Recent Developments... (last updated 11/25/96) [INLINE]

- Amex Card Gets Smart For Travellers
- Canadian Internet Banking - A Summary
- Verifone Makes E-Commerce An Everyday Affair
- Internet Banking - The Next Killer App
- Canada's Smart Card Deal With China
- Europe Leads The World In Smart Card Usage
- CyberCoin & Internet Wallet
- TD "To Do" Banking On The Internet
- Digital Signature - Your ID on The WWW
- The 'Net Gives New Life To EDI
- Kingston Gets Smart With EXACT Card
- Visa Cash To be Used In Vancouver's AIDS Conference
- Verifone 'SET' To Kick Start Internet Commerce
- Mondex Turns Guelph, ON Into Smart Card City
- From Fur Trade To Cyber Trade, Hudson Bay Goes Again
- Indonesia To Install 10000+ Smart Card Phones
- EDI To Merge With Internet
- CyberCash Net Commerce A Reality In Canada
- Bankers Facing Real Changes In Virtual World
- Mondex Cards Give Visa Cash A Run For Their Money
- Visa Cash Micropayment Cards To Be Used In Hong Kong
- IBM Builds NetCommerce Product
- Joint Electronic Payment Initiative (JEPI)
- New Products For Internet Security and Authentication
- New WWW Formats For Grocery Traders
- US Banks To Test 'Smart Cards' In NYC
- Smart Cards & Internet Point-Of-Sale Devices
- Wal-Mart Goes Online With Microsoft
- Visa & Mastercard Set For Internet Commerce
- Netscape & Verifone's Internet Payment Pact
- IRS says "See you on the web!"
- Electronic commerce over the WWW

ELECTRONIC COMMERCE RESOURCES

Electronic Commerce World Institute

Used to be known as EDI World Institute, this international body based in Montreal, Canada has changed its name in November 1995. They have one of the most complete references on the subject of electronic commerce and EDI. http://www.ediwi.ca:6900/

Center for Electronic Commerce

A U.S. institution located in Ann Arbor, Michigan aimed at promoting technologies for use in electronic commerce. They are responsible for developing methodologies for industry adoption. http://www.iti.org:80/cec/

Fairfax Electronic Commerce Resource Centre

http://www.ecrc.gmu.edu/

Federal Electronic Commerce for Acquisition Team (ECAT)

http://snad.ncsl.nist.gov/dartg/dartg-projects.html#sec-registry

The DOD (Department of Defence) Electronic Commerce Office

The Federal Acquisition Streamlining Act of 1994 (PL 103-355) requires the broad use of Electronic Commerce and Electronic Data Interchange (EC/EDI) by Federal agencies. This Web site has information on EC/EDI programs and DoD resources. http://www.acq.osd.mil/ec/ecic_hpg.html

Thomas Ho's Electronic Commerce Guide

http://www.engr.iupui.edu/~ho/interests/commmenu.html

U.S. General Services Administration

If you are planning to sell to the U.S. Government, this web site is for you. http://www.fss.gsa.gov/

U.S. Government Small Business Administration

Gopher and telnet service provided by the Small Business Administration (SBA) of the U.S. Federal Government. It offers information about general business topics. gopher://www.sbaonline.sba.gov/1

The Electronic Commerce Association

ECA is a London UK based organization whose purpose is to promote better business practices through the use of electronic commerce. http://www.eca.org.uk

The Canadian Youth Business Foundation

Dedicated to young people in Canada eager to be entrepreneurs. http://www.cybf.ca

ELECTRONIC DATA INTERCHANGE (EDI)

Electronic Data Interchange Council of Canada (EDICC)

http://www.edicc.ca/edicc/

The BT EDI Research Centre (UK)

Established in 1987 and funded by the British Telecommunications plc to foster the development and implementation of EDI within the transportation and other industries. http://info.cardiff.ac.uk/uwcc/masts/parfett/edihome.html

Federal EDI -US Government agency

http://snad.ncsl.nist.gov/dartg/edi/fededi.html

UN Agencies:

gopher://info.itu.ch/11/.1/UNagencies

ELECTRONIC CASH OR DIGITAL TOKENS

XIWT- Cross Industry Working Team

A U.S. industry coalition dedicated to define technical requirements for formulating a powerful and sustainable national information infrastructure (NII). This web site has excellent refernce material on electronic cash or digital tokens. http://www.cnri.reston.va.us:3000/XIWT/documents/dig_cash_doc/ToC.html

Carnegie Mellon University - NetBill

CMU has been working on electronic payment methods over the Internet since 1991. They are currently working with VISA on the NetBill project. http://www.ini.cmu.edu/netbill/

INTERNATIONAL STANDARDS ORGANIZATIONS

Internet Engineering Task Force (IETF)

Mentioning any technology development reated to the Internet without mentioning the IETF would not be right. So here it is. http://www.ietf.cnri.reston.va.us/

Data Interchange Standards Association (DISA)

http://www.disa.org/

American National Standards Institute (ANSI)

Has the most up-to-date information about standards activities around the world. Fee-based membership required to access database. http://www.ansi.org

The International Organization for Standardization

http://www.iso.ch/

[LINK] TOP

Copyright 1996. &copy; COSMO21/Cosmotek Interactive Technology Inc.
All rights reserved.

http://www.counterpane.com/whycrypto.html.

Why Cryptography is Harder than it Looks

From e-mail to cellular communications, from secure Web access to digital cash, cryptography is an essential part of today's information systems. Cryptography helps provide accountability, fairness, accuracy, and confidentiality. It can prevent fraud in electronic commerce and assure the validity of financial transactions. It can prove your identity or protect your anonymity. It can keep vandals from altering your Web page and prevent industrial competitors from reading your confidential documents. And in the future, as commerce and communications continue to move to computer networks, cryptography will become more and more vital.

But the cryptography now on the market doesn't provide the level of security it advertises. Most systems are not designed and implemented in concert with cryptographers, but by engineers who thought of cryptography as just another component. It's not. You can't make systems secure by tacking on cryptography as an afterthought. You have to know what you are doing every step of the way, from conception through installation.

Billions of dollars are spent on computer security, and most of it is wasted on insecure products. After all, weak cryptography looks the same on the shelf as strong cryptography. Two e-mail encryption products may have almost the same user interface, yet one is secure while the other permits eavesdropping. A comparison chart may suggest that two programs have similar features, although one has gaping security holes that the other doesn't. An experienced cryptographer can tell the difference. So can a thief.

Present-day computer security is a house of cards; it may stand for now, but it can't last. Many insecure products have not yet been broken because they are still in their infancy. But when these products are widely used, they will become tempting targets for criminals. The press will publicize the attacks, undermining public confidence in these systems. Ultimately, products will win or lose in the marketplace depending on the strength of their security.

Threats to computer systems

Every form of commerce ever invented has been subject to fraud, from rigged scales in a farmers' market to counterfeit currency to phony invoices. Electronic commerce schemes will also face fraud, through forgery, misrepresentation, denial of service, and cheating. In fact, computerization makes the risks even greater, by allowing attacks that are impossible against non-automated systems. A thief can make a living skimming a penny from every Visa cardholder. You can't walk the streets wearing a mask of someone else's face, but in the digital world it is easy to impersonate others. Only strong cryptography can protect against these attacks.

Privacy violations are another threat. Some attacks on privacy are targeted: a member of the press tries to read a public figure's e-mail, or a company tries to intercept a competitor's communications. Others are broad data-harvesting attacks, searching a sea of data for interesting information: a list of rich widows, AZT users, or people who view a particular Web page.

Electronic vandalism is an increasingly serious problem. Computer vandals have already graffitied the CIA's web page, mail-bombed Internet providers, and canceled thousands of newsgroup messages. And of course, vandals and thieves routinely break into networked computer systems. When security safeguards aren't adequate, trespassers run little risk of getting caught.

Attackers don't follow rules; they cheat. They can attack a system using techniques the designers never thought of. Art thieves have burgled homes by cutting through the walls with a chain saw. Home security systems, no matter how expensive and sophisticated, won't stand a chance against this attack. Computer thieves come through the walls too. They steal technical data, bribe insiders, modify software, and collude. They take advantage of technologies newer than the system, and even invent new mathematics to attack the system with.

The odds favor the attacker. Bad guys have more to gain by examining a system than good guys. Defenders have to protect against every possible vulnerability, but an attacker only has to find one security flaw to compromise the whole system.

What cryptography can and can't do

No one can guarantee 100% security. But we can work toward 100% risk acceptance. Fraud exists in current commerce systems: cash can be counterfeited, checks altered, credit card numbers stolen. Yet these systems are still successful because the benefits and conveniences outweigh the losses. Privacy systems wall safes, door locks, curtains are not perfect, but they're often good enough. A good cryptographic system strikes a balance between what is possible and what is acceptable.

Strong cryptography can withstand targeted attacks up to a point the point at which it becomes easier to get the information some other way. A computer encryption program, no matter how good, will not prevent an attacker from going through someone's garbage. But it can prevent data-harvesting attacks absolutely; no attacker can go through enough trash to find every AZT user in the country. And it can protect communications against non-invasive attacks: it's one thing to tap a phone line from the safety of the telephone central office, but quite another to break into someone's house to install a bug.

The good news about cryptography is that we already have the algorithms and protocols we need to secure our systems. The bad news is that that was the easy part; implementing the protocols successfully requires considerable expertise. The areas of security that interact with people key management, human/computer interface security, access control often defy analysis. And the disciplines of public-key infrastructure, software security, computer security, network security, and tamper-resistant hardware design are very poorly understood.

Companies often get the easy part wrong, and implement insecure algorithms and protocols. But even so, practical cryptography is rarely broken through the mathematics; other parts of systems are much easier to break. The best protocol ever invented can fall to an easy attack if no one pays attention to the more complex and subtle implementation issues. Netscape's security fell to a bug in the random-number generator. Flaws can be anywhere: the threat model, the system design, the software or hardware implementation, the system management. Security is a chain, and a single weak link can break the entire system. Fatal bugs may be far removed from the security portion of the software; a design decision that has nothing to do with security can nonetheless create a security flaw.

Once you find a security flaw, you can fix it. But finding the flaws in a product can be incredibly difficult. Security is different from any other design requirement, because functionality does not equal quality. If a word processor prints successfully, you know that the print function works. Security is different; just because a safe recognizes the correct combination does not mean that its contents are secure from a safecracker. No amount of general beta testing will reveal a security flaw, and there's no test possible that can prove the absence of flaws.

Threat models

A good design starts with a threat model: what the system is designed to protect, from whom, and for how long. The threat model must take the entire system into account not just the data to be protected, but the people who will use the system and how they will use it. What motivates the attackers? Must attacks be prevented, or can they just be detected? If the worst happens and one of the fundamental security assumptions of a system is broken, what kind of disaster recovery is possible? The answers to these questions can't be standardized; they're different for every system. Too often, designers don't take the time to build accurate threat models or analyze the real risks.

Threat models allow both product designers and consumers to determine what security measures they need. Does it makes sense to encrypt your hard drive if you don't put your files in a safe? How can someone inside the company defraud the commerce system? How much would it cost to defeat the tamper-resistance on the smart card? You can't design a secure system unless you understand what it has to be secure against.

System design

Design work is the mainstay of the science of cryptography, and it is very specialized. Cryptography blends several areas of mathematics: number theory, complexity theory, information theory, probability theory, abstract algebra, and formal analysis, among others. Few can do the science properly, and a little knowledge is a dangerous thing: inexperienced cryptographers almost always design flawed systems. Good cryptographers know that nothing substitutes for extensive peer review and years of analysis. Quality systems use published and well-understood algorithms and protocols; using unpublished or unproven elements in a design is risky at best.

Cryptographic system design is also an art. A designer must strike a balance between security and accessibility, anonymity and accountability, privacy and availability. Science alone cannot prove security; only experience, and the intuition born of experience, can help the cryptographer design secure systems and find flaws in existing designs.

Implementation

There is an enormous difference between a mathematical algorithm and its concrete implementation in hardware or software. Cryptographic system designs are fragile. Just because a protocol is logically secure doesn't mean it will stay secure when a designer starts defining message structures and passing bits around. Close isn't close enough; these systems must be implemented exactly, perfectly, or they will fail. A poorly-designed user interface can make a hard-drive encryption program completely insecure. A false reliance on tamper-resistant hardware can render an electronic commerce system all but useless. Since these mistakes aren't apparent in testing, they end up in finished products. Many flaws in implementation cannot be studied in the scientific literature because they are not technically interesting. That's why they crop up in product after product. Under pressure from budgets and deadlines, implementers use bad random-number generators, don't check properly for error conditions, and leave secret information in swap files. The only way to learn how to prevent these flaws is to make and break systems, again and again.

Cryptography for people

In the end, many security systems are broken by the people who use them. Most fraud against commerce systems is perpetrated by insiders. Honest users cause problems because they usually don't care about security. They want simplicity, convenience, and compatibility with existing (insecure) systems. They choose bad passwords, write them down, give friends and relatives their private keys, leave computers logged in, and so on. It's hard to sell door locks to people who don't want to be bothered with keys. A well-designed system must take people into account.

Often the hardest part of cryptography is getting people to use it. It's hard to convince consumers that their financial privacy is important when they are willing to leave a detailed purchase record in exchange for one thousandth of a free trip to Hawaii. It's hard to build a system that provides strong authentication on top of systems that can be penetrated by knowing someone's mother's maiden name. Security is routinely bypassed by store clerks, senior executives, and anyone else who just needs to get the job done. Only when cryptography is designed with careful consideration of users' needs and then smoothly integrated, can it protect their systems, resources, and data.

The state of security

Right now, users have no good way of comparing secure systems. Computer magazines compare security products by listing their features, not by evaluating their security. Marketing literature makes claims that are just not true; a competing product that is more secure and more expensive will only fare worse in the market. People rely on the government to look out for their safety and security in areas where they lack the knowledge to make evaluations food packaging, aviation, medicine. But for cryptography, the U.S. government is doing just the opposite.

When an airplane crashes, there are inquiries, analyses, and reports. Information is widely disseminated, and everyone learns from the failure. You can read a complete record of airline accidents from the beginning of commercial aviation. When a bank's electronic commerce system is breached and defrauded, it's usually covered up. If it does make the newspapers, details are omitted. No one analyzes the attack; no one learns from the mistake. The bank tries to patch things in secret, hoping that the public won't lose confidence in a system that deserves no confidence. In the long run, secrecy paves the way for more serious breaches.

Laws are no substitute for engineering. The U.S. cellular phone industry has lobbied for protective laws, instead of spending the money to fix what should have been designed corectly the first time. It's no longer good enough to install security patches in response to attacks. Computer systems move too quickly; a security flaw can be described on the Internet and exploited by thousands. Today's systems must anticipate future attacks. Any comprehensive system--whether for authenticated communications, secure data storage, or electronic commerce--is likely to remain in use for five years or more. It must be able to withstand the future: smarter attackers, more computational power, and greater incentives to subvert a widespread system. There won't be time to upgrade them in the field.

History has taught us: never underestimate the amount of money, time, and effort someone will expend to thwart a security system. It's always better to assume the worst. Assume your adversaries are better than they are. Assume science and technology will soon be able to do things they cannot yet. Give yourself a margin for error. Give yourself more security than you need today. When the unexpected happens, you'll be glad you did.

Home page - Counterpane - Applied Cryptography - E-Mail Security - Crypto-Links
Bruce Schneier Bio - Blowfish - Publications - Site Index - Contact Counterpane

---

Bruce Scheneier
Copyright © 1995 Counterpane Systems All rights reserved.
Revised: November 02, 1996.

How private are "private" electronic payment systems?

I did a bit of net.research recently to play the "Cash" claim on the Information Futures market at <URL:http://if.arc.ab.ca/~jamesm/IF/>. This claim has to do with the degree of success of an anonymous payment system on the Internet by the end of 1996. Information Futures is a game (using play money) modeled after the commodity futures markets, but trading on ideas instead of pork bellies and soy beans.

I discovered many more e-money schemes than I thought there would be, but *all* of them failed the specific tests of the claim, namely that both purchaser and seller need to be anonymous. All methods I've come across allow the identity of at least one of the parties to at least theoretically be determined, including both Mondex and Digicash, which I previously thought viable (DigiCash allows a seller to be identified by a bank).

However, in addition to DigiCash, both Brand's Cash and AT&T Anonymous Credit Cards may allow purchaser anonymity. This evaluation is from reading the scarce information I came across which may have left out important details. One or more of the methods at the end of the list (which I did not verify) may qualify as well.

Many of the schemes I came across bill themselves as providing "privacy", yet keep logs of purchasers and the vendors they frequent. Several methods, however, (such as NetCash(SM)) provide practical anonymity in which the identities of parties cannot be determined without collusion with a central "bank" or clearinghouse of some sort. If this "bank" has a policy of not logging certain transactions which could be used to track identities, the result is effective anonymity. DigiCash is the only method I could find that can be shown mathematically to allow a buyer to be perfectly anonymous.

Following is a summary of all the digital payment methods I've come across. The information on anonymity and implementation status is MY OWN OPINION based on information I read (actually, skimmed, in most cases) on the Web. Please correct me if I'm wrong. I'll keep an HTML version of this page up at <URL:http://vanbc.wimsey.com/~danf/emoney-anon.html> for the time being.

DigiCash <http://www.digicash.com>
- purchaser is anonymous unless he double spends, seller's identity can be found by collusion with bank
- implementation: in full operation since late 1995

CAFE <http://www.digicash.com>
- purchaser is anonymous, seller's identity can be found by collusion with bank
- implementation: unknown

Brand's Cash <http://ganges.cs.tcd.ie/mepeirce/Project/Mlists/brands.html>
- purchaser is anonymous unless he double spends
- implementation: unknown

NetBank/NetCash <http://www.teleport.com/~netcash/> (not to be confused with the NetCheque/NetCash(SM) system below)
- anonymity: transactions between purchaser and seller can be linked in most cases
- implementation: in full operation

CyberCash <http://www.cybercash.com>
<http://ganges.cs.tcd.ie/mepeirce/Project/Press/cybernote.html>
- anonymity: CyberCash server can and must link transactions between purchaser and seller (via credit card)
- implementation: in full operation

First Virtual <http://www.fv.com>
http://www.c2.org/nofv/
- transactions between purchaser and seller are linked by central server
- implementation: in full operation

Credit card number via http + SSL (e.g. Netscape)
- neither purchaser nor seller is anonymous
- implementation: in full operation

NetCheque(SM) & NetCash(SM)
<http://nii-server.isi.edu/info/NetCheque/>
<http://ganges.cs.tcd.ie/mepeirce/Project/Press/nc.html>
<http://ganges.cs.tcd.ie/mepeirce/Project/Press/net2a4.ps>
- anonymity: NetCash(SM) identity of purchaser can be determined by collusion among all parties (may be extended to provide perfect anonymity to purchaser); NetCheque(SM) is not anonymous
- implementation: NetCheque in test phase; NetCash not yet implemented

Mondex <http://www.mondex.com/mondex/>
- neither purchaser nor seller is anonymous
(see <http://ganges.cs.tcd.ie/mepeirce/Project/Press/monpriv.html>)
- implementation: smart card trial in progress in UK; Internet status unknown

NetBill <http://www.ini.cmu.edu/netbill/>
<http://www.visa.com/cgi-bin/vee/vw/news/PRelco021595.html?2+0>
- anonymity: "Protects the privacy of the transaction"; appears that in actuality, central NetBill server can and must link purchaser and seller in a transaction
- implementation: trial should be running around this time

Millicent
<http://www.research.digital.com/SRC/millicent/papers/millicent-w3c4/millicent.html>
- neither purchaser nor seller is anonymous
- implementation: tests were planned for the end of 1995

First Bank of Internet (FBOI)
<http://ganges.cs.tcd.ie/mepeirce/Project/Press/fboi.html>
- transactions between purchaser and seller are linked by central server
- implementation: in full operation

NetChex <http://www.netchex.com>

<http://ganges.cs.tcd.ie/mepeirce/Project/Press/netchex.html>
- neither purchaser nor seller is anonymous
- implementation: unknown

CyberCard <http://netmarket.com/nm/pages/cuc/sid=LMAmZEJeI7>
- anonymity: unknown
- implementation: unknown

iKP (IBM)
<http://www.zurich.ibm.ch/Technology/Security/extern/ecommerce/>
- anonymity: transactions linked between purchaser and seller via credit card in standard implementation
- implementation: unknown

Micropayments (Hewlett Packard)
<http://www.hpl.hp.co.uk/projects/vishnu/main.html>
- neither purchaser nor seller is anonymous
- implementation: unknown

Anonymous Credit Cards (AT&T)
<ftp://research.att.com/dist/anoncc/anoncc.ps.Z>
- anonymity: transactions can be linked between purchaser and seller by collusion; anonymous debit card can be anonymous for purchaser, but not seller
- implementation: unknown

PayMe (based on NetCash)
<http://www.w3.org/pub/Conferences/WWW4/Papers/228/>
- anonymity: transactions between purchaser and seller can be linked if currency servers log all transactions
- implementation: unknown

The following proposals I haven't evaluated:

SNPP (MIT)
<http://ganges.cs.tcd.ie/mepeirce/Project/Oninternet/snpp-paper.ps.Z>
- "does not provide anonymity" -- Low & Maxemchuk
- implementation: unknown

MarketNet (BankNet)

NetPay (Boston Automation)

STT (Visa & Microsoft)
<http://www.microsoft.com/intdev/inttech/wire15dx.htm>

SVC (VISA)

Web Payment System (Open Market)

Merita Solo Payment system

Anonymous Internet Mercantile Protocol

CARI (Collect All Relevant Information)

Checkfree

Clickshare

Commercenet

Downtown Anywhere

FSTC Electronic Check Project

Globe ID

LETSystems

Magic Money

Micro Payment Transfer Protocol (MPTP)

Online Check Systems

PayWord and MicroMint

Secure Electronic Payment Protocol (SEPP) (Mastercard)

Ziplock

Lists of digital cash resources:

<http://ganges.cs.tcd.ie/mepeirce/Project/oninternet.html>
<http://ganges.cs.tcd.ie/mepeirce/Project/press.html>
<http://www.yahoo.com/Business_and_Economy/Companies/Financial_Services/Digital_Money/>
<http://www.ini.cmu.edu/NETBILL/commerce.html>
<http://www.w3.org/pub/WWW/Payments/>
<http://www.w3.org/pub/WWW/Payments/Talks/w3c_security_10-jul-95/talk.html>

[LINK] back to Dan's personal home page [LINK] Sponsored by Wimsey

---

dan@fch.wimsey.bc.ca

Last updated 1996-04-19

[Help] [Welcome] [Boolean] [Advanced] [Sales]

[USPTO] [CNIDR] [Previous Patent] [Next Patent] [Back to List]

United States Patent 5,511,121 YacobiApr. 23, 1996

Efficient electronic money

Inventors: Yacobi; Yacov (Berkeley Heights, NJ). Assignee: Bell Communications Research, Inc. (Morristown, NJ). Appl. No.: 303,048 Filed: Sept. 8, 1994

Related U.S. Application Data Continuation-in-part of Ser No. 201,106, Feb. 23, 1994. Intl. Cl.:H04K 1/00 U.S. Cl.:380/ 24; 380/ 23; 380/ 30 Current U.S. Class:380/24 Field of Search:380/23, 24, 25, 28, 30

References Cited [Referenced By:]

U.S. Patent Documents 4,977,595Dec., 1990Ohta et al. 380/ 24 5,199,070Mar., 1993Matsuzaki et al. 380/ 30 5,222,140Jun., 1993Beller et al. 380/ 30 5,224,162Jun., 1993Okamoto et al. 380/ 24 5,299,263Mar., 1994Beller et al. 380/ 30

Foreign Patent Documents 0139313May, 1985EP 380/ 24 0348812Jun., 1989EP 380/ 24

Other References

W. Diffie et al., "New Directions in Cryptography", IEEE Transactions on Information Theory, 1976, vol. IT-22, pp. 644-654.

R. L. Rivest et al., "A Method for Obtaining Digital Signatures and Public-Key Cryptosystems," Communications of the ACM, 1978, vol. 21, pp. 120-126.

D. Chaum et al., "Untraceable Electronic Cash", Proceedings of Crypto 1988, pp. 1-8.

D. Chaum, "Achieving Electronic Privacy," 1992, Scientific American, pp. 96-101.

S. Brands, "Electronic Cash Systems Based On The Representation Problem in Groups of Prime Order," Proceedings of Crypto '93, Santa Barbara, CA., pp. 26.1-26.15.

S. Evan et al., "Electronic Wallet," Proceedings of the 1984 International Zurich seminar on Digital Communications, Santa Barabara, CA, pp. K 3.1-K 3.2.

T. Elgamal, "A Public Key Cryptosystem and a Signature Scheme Based on Discrete Logarithms," IEEE Transactions on Information Theory, 1985, vol. IT-31, pp. 469-472.

O. Goldreich et al., "Proofs that Yield Nothing But Their Validity or All Languages in NP Have Zero-Knowledge Proof Systems," Journal of the Association for Computing Machinery, 1991, vol. 38, pp. 691-729.

S. Goldwasser et al., "The Knowledge Complexity of Interactive

Proof Systems," SIAM Journal on Computing, 1989, vol. 18, pp. 186-208.

E. F. Brickell et al. "Gradual and Verifiable Release of a Secret," Proc. Crypto '87, pp. 156-166.

M. O. Rabin, "Digitalized Signatures and Public-Key Functions as Intractable as Factorization," MIT Laboratory For Computer Science, Jan. 1979, pp. 1-16.

Primary Examiner: Cain; David C.
Attorney, Agent or Firm: Suchyta; Leonard Charles, Falk; James W.

Abstract

A unique electronic cash system protects the privacy of users in legitimate transactions while at the same time enabling the detection of a double spender of the same electronic coin. The electronic cash system takes advantage of a unique property of El Gamal signatures to achieve these results.

51 Claims, 9 Drawing Figures

[USPTO] [CNIDR]

[Help] [Welcome] [Boolean] [Advanced] [Sales]

[USPTO] [CNIDR] [Previous Patent] [Next Patent] [Back to List]

United States Patent 5,455,407 Rosen* Oct. 3, 1995

Electronic-monetary system

Inventors: Rosen; Sholom S. (New York, NY). Assignee: Citibank, N.A. (New York, NY). [*] Notice: The portion of the term of this patent subsequent to Sept. 26, 2012 has been disclaimed. Appl. No.: 378,955 Filed: Jan. 27, 1995

Related U.S. Application Data Division of Ser No. 794,112, Nov. 15, 1991. Intl. Cl.:G06K 5/00 U.S. Cl.:235/380; 235/379; 364/408; 902/ 24 Current U.S. Class:235/380 Field of Search:235/380, 379; 902/24; 364/406

References Cited [Referenced By:]

U.S. Patent Documents 3,559,175Jan., 1971Pomeroy 3,573,747Apr., 1971Adams et al. 3,749,887Jul., 1973Giuliani 3,852,571Dec., 1974Hall et al. 3,906,460Sept., 1975Halpern 3,932,730Jan., 1976Ambrosio 3,934,122Jan., 1976Riccitelli 3,937,925Feb., 1976Boothroyd 3,971,916Jul., 1976Moreno 4,001,550Jan., 1977Schatz 4,007,355Feb., 1977Moreno 4,053,735Oct., 1977Fondos 4,120,452Oct., 1978Kimura et al. 4,172,552Oct., 1979Case 4,179,064Dec., 1979Yoshioka et al. 4,214,230Jul., 1980Fak et al. 4,218,582Aug., 1980Hellman et al. 4,224,666Sept., 1980Giraud 4,256,955Mar., 1981Giraud et al. 4,270,042May, 1981Case 4,277,837Jul., 1981Stuckert 4,302,810Nov., 1981Bouricius et al. 4,305,059Dec., 1981Benton 4,320,387Mar., 1982Powell 4,321,672Mar., 1982Braun et al. 4,341,951Jul., 1982Benton 4,404,649Sept., 1983Nunley et al. 4,405,829Sept., 1983Rivest et al. 4,442,345Apr., 1984Mollier et al. 4,443,027Apr., 1984McNeeley et al. 4,453,074Jun., 1984Weinstein 4,454,414Jun., 1984Benton 4,460,965Jul., 1984Trehn et al. 4,467,139Aug., 1984Mollier 4,498,000Feb., 1985Decavele et al. 4,511,970Apr., 1985Okano et al. 4,523,087Jun., 1985Benton 4,523,297Jun., 1985Ugon et al. 4,529,870Jul., 1985Chaum 4,536,647Aug., 1985Atalla et al. 4,549,075Oct., 1985Saada et al. 4,614,861Sept., 1986Pavlov et al. 4,625,276Nov., 1986Benton et al. 4,629,872Dec., 1986Hallberg 4,630,201Dec., 1986White 4,634,845Jan., 1987Hale et al. 4,642,768Feb., 1987Roberts 4,650,978Mar., 1987Hudson et al. 4,667,088May, 1987Kramer et al. 4,673,802Jun., 1987Ohmae et al. 4,689,478Aug., 1987Hale et al. 4,692,601Sept., 1987Nakano 4,697,073Sept., 1987Hara 4,705,211Nov., 1987Honda et al. 4,722,055Jan., 1988Roberts 4,727,243Feb., 1988Savar 4,727,244Feb., 1988Nakano et al. 4,734,568Mar., 1988Watanabe 4,736,094Apr., 1988Yoshida 4,742,215May, 1988Daughters et al. 4,748,668May, 1988Shamir et al. 4,750,119Jun., 1988Cohen et al. 4,751,640Jun., 1988Lucas et al. 4,752,676Jun., 1988Leonard et al. 4,752,877Jun., 1988Roberts et al. 4,757,185Jul., 1988Onishi 4,759,064Jul., 1988Chaum 4,766,539Aug., 1988Fox 4,767,920Aug., 1988Kitta et al. 4,799,156Jan., 1989Shavit et al. 4,823,264Apr., 1989Deming 4,825,052Apr., 1989Chemin et al. 4,827,112May, 1989Yoshino et al. 4,837,422Jun., 1989Dethloff et al. 4,839,504Jun., 1989Nakano 4,877,947Oct., 1989Mori 4,879,747Nov., 1989Leighton et al. 4,906,828Mar., 1990Halpern 4,914,698Apr., 1990Chaum 4,926,480May, 1990Chaum 4,949,380Aug., 1990Chaum 4,962,530Oct., 1990Cairns 4,964,164Oct., 1990Fiat 4,968,873Nov., 1990Dethloff et al. 4,973,828Nov., 1990Naruse et al. 4,977,595Dec., 1990Ohta et al. 4,985,833Jan., 1991Oncken 4,987,593Jan., 1991Chaum 4,991,210Feb., 1991Chaum 4,992,646Feb., 1991Collin 4,995,081Feb., 1991Leighton et al. 4,996,711Feb., 1991Chaum 5,191,193Mar., 1993LeRoux

Foreign Patent Documents 172670A2Feb., 1986EP 391261B1Mar., 1990EP 421808A3Apr., 1991EP 8303018Sept., 1983WO 9110669lOct., 1991WO 9308545Apr., 1993WO

Other References

David Chaum, Achieving Electronic Privacy, Scientific American, Aug. 1992.

Dancoin Ltd., The Company, Danmont A/S month missing 1991.

O'Reilly, Ireland's Pocket Revolution: The Micro That Replaces Cash, Cheques, And Cards, Retail Banker International, Feb. 20, 1984, at 4.

Nakamoto, Japanese Take To The Top The Prepaid Plastic Card Business, Financial Times, Nov. 17, 1988, at 7.

Rowe, Au Revoir Le Cash?, Banking Technology, Jul.-Aug. 1991, at 46.

Okamoto and Ohta, Universal Electronic Cash, Cryptography Symposium month missing (1991).

Article 4A, Uniform Commerical Code, Callaghan & Company, dated Apr. 1990.

"Security Without Identification: Transaction Systems To Make Big Brother Obsolete" Chaum, D., Communication of the ACM, 28:10, Oct. 1985.

"Disposable Zero-Knowledge Authentications and Their Applications To Untraceable Electronis Cash", Okamoto, T. et al., 481-496, undated month & year missing.

1984 International Zurich Seminar On Digital Communications, Electronic Wallet, S. Even, O. Goldreich, Y. Yacobi, month missing 1984.

Privacy Protected Payments Unconditional Payer and/or Payee Untraceability, D. Chaum, Smart Card 2000, month missing 1989.

Security Without Identification: Card Computers To Make Big Brother Obsolete, D. Chaum, month missing 1987.

Untraceable Electronic Cash, D. Chaum, et al. month and year missing.

Thomas M. Atwood, The case for object-oriented databases, IEEE Spectrum, Feb. 1991.

David Chaum, Online Cash Checks, Centre for Mathematics and Computer Science, Amsterdam month and year missing.

Primary Examiner: Shepperd; John
Assistant Examiner: Lee; Michael G.
Attorney, Agent or Firm: Dowling; Thomas P., Bromberg; Laurence J.

Abstract

An improved monetary system using electronic media to exchange economic value securely and reliably. The invention provides a complete monetary system having electronic money that is interchangeable with conventional paper money comprising (1) issuing banks or financial institutions that are coupled to a money generator device for generating and issuing to subscribing customers electronic money including electronic currency backed by demand deposits, or electronic credit authorizations; (2) correspondent banks that accept and distribute the electronic money; (3) a plurality of transaction devices that are used by subscribers for storing electronic money, for performing money transactions with the on-line systems of the participating banks or for exchanging electronic money with other like transaction devices; (4) teller devices, associated with the issuing and correspondent banks, for process handling and interfacing the transaction devices to the issuing and correspondent banks, and for interfacing between the issuing and correspondent banks themselves; (5) a security arrangement for maintaining the integrity of the system; and (6) reconciliation and clearing processes to monitor and balance the monetary system.

9 Claims, 69 Drawing Figures

[USPTO] [CNIDR]

[Help] [Welcome] [Boolean] [Advanced] [Sales]

[USPTO] [CNIDR] [Previous Patent] [Next Patent] [Back to List]

United States Patent 5,453,601 RosenSept. 26, 1995

Electronic-monetary system

Inventors: Rosen; Sholom S. (New York, NY). Assignee: Citibank, N.A. (New York, NY). Appl. No.: 794,112 Filed: Nov. 15, 1991 Intl. Cl.:G06F 17/60, G06G 7/52 U.S. Cl.:235/379; 364/408; 902/ 24 Current U.S. Class:235/379 Field of Search:235/379; 364/406, 408; 902/24

References Cited [Referenced By:]

U.S. Patent Documents 3,559,175Jan., 1971Pomeroy 3,573,747Apr., 1971Adams 3,749,887Jul., 1973Giuliani 3,852,571Dec., 1974Hall et al. 3,906,460Sept., 1975Halpern 3,932,730Feb., 1976Ambrosio 3,934,122Jan., 1976Riccitelli 3,937,925Feb., 1976Boothroyd 4,001,550Jan., 1977Schatz 4,007,355Feb., 1977Moreno 4,053,735Oct., 1977Foados 4,120,452Oct., 1978Kimura et al. 4,172,552Oct., 1979Case et al. 4,179,064Dec., 1979Yoshioka et al. 4,214,230Jul., 1980Fak et al. 4,218,582Aug., 1980Hellman et al. 4,224,666Sept., 1980Giraud 4,256,955Mar., 1981Giraud et al. 4,270,042May, 1981Case 4,277,837Jul., 1981Stuckert 4,302,810Nov., 1981Bouricius et al. 4,305,059Dec., 1981Benton 4,320,387Mar., 1982Powell 4,321,672Mar., 1982Braun et al. 4,341,951Jul., 1982Benton 4,404,649Sept., 1983Nunley et al. 4,405,829Sept., 1983Rivest et al. 4,442,345Apr., 1984Mollier et al. 4,443,027Apr., 1984McNeely et al. 4,453,074Jun., 1984Weinstein 4,454,414Jun., 1984Benton 4,460,965Jul., 1984Trehn et al. 4,467,139Aug., 1984Mollier 4,498,000Feb., 1985Decavele et al. 4,511,970Apr., 1985Okano et al. 4,523,087Jun., 1985Benton 4,523,297Jun., 1985Ugon et al. 4,529,870Jul., 1985Chaum 4,536,647Aug., 1985Atalla et al. 4,549,075Oct., 1985Saada et al. 4,597,046Jun., 1986Musmanno et al. 4,614,861Sept., 1986Pavlov et al. 4,625,276Nov., 1986Benton et al. 4,629,872Dec., 1986Hallberg 4,630,201Dec., 1986White 4,634,845Jan., 1987Hale et al. 4,642,768Feb., 1987Roberts 4,650,978Mar., 1987Hudson et al. 4,673,802Jun., 1987Ohmae et al. 4,677,088May, 1987Kramer et al. 4,689,478Aug., 1987Hale et al. 4,692,601Sept., 1987Nakano 4,697,073Sept., 1987Hara 4,705,211Nov., 1987Honda et al. 4,722,055Jan., 1988Roberts 4,727,243Feb., 1988Savar 4,727,244Feb., 1988Nakano et al. 4,734,568Mar., 1988Watanabe 4,736,094Apr., 1988Yoshida 4,742,215May, 1988Daughters et al. 4,748,668May, 1988Shamir et al. 4,750,119Jun., 1988Cohen et al. 4,751,640Jun., 1968Lucas et al. 4,752,676Jun., 1988Leonard et al. 4,752,877Jun., 1988Roberts et al. 4,757,185Jul., 1988Onishi 4,759,064Jul., 1988Chaum 4,766,539Aug., 1988Fox 4,767,920Aug., 1988Kitta et al. 4,799,156Jan., 1989Shavit et al. 4,823,264Apr., 1989Deming 4,825,052Apr., 1989Chemin et al. 4,827,112May, 1989Yoshino et al. 4,837,422Jun., 1989Dethloff et al. 4,877,947Oct., 1989Mori 4,879,747Nov., 1989Leighton et al. 4,906,828Mar., 1990Halpern 4,914,698Apr., 1990Chaum 4,949,380Aug., 1990Chaum 4,962,530Oct., 1990Cairns 4,964,164Oct., 1990Fiat 4,968,873Nov., 1990Dethloff et al. 4,977,595Dec., 1990Ohta et al. 4,985,833Jan., 1991Oncken 4,987,593Jan., 1991Chaum 4,991,210Feb., 1991Chaum 4,992,646Feb., 1991Collin 4,995,081Feb., 1991Leighton et al. 4,996,711Feb., 1991Chaum

Foreign Patent Documents 391261B1Oct., 1990EP WO9116691Oct., 1991WO WO9308545Apr., 1993WO

Other References

"Security Without Identification: Transaction Systems To Make Big Brother Obsolete" Chaum, D., Communications of the ACM, 28:10, Oct. 1985.

Article 4A, Uniform Commercial Code, Callaghan & Company, dated Apr. 1990.

"Disposable Zero-Knowledge Authentications and Their Applications to Untraceable Electronic Cash", Okamoto, T. et al., 481-496, undated.

1984 International Zurich Seminar On Digital Communications, Electronic Wallet, S. Even, O. Goldreich, Y. Yacobi, 1984.

Thomas M. Atwood, The case for object-oriented databases, IEEE Spectrum, Feb. 1991.

David Chaum, Online Cash Checks, Centre for Mathematics and Computer Science, Amsterdam.

David Chaum, Achieving Electronic Privacy, Scientific American, Aug. 1992.

Dancoin Ltd., The Company, Danmont A/S 1991.

O'Reilly, Ireland's Pocket Revolution: The Micro That Replaces Cash, Cheques, And Cards, Retail Banker International, Feb. 20, 1984, at 4.

Nakamoto, Japanese Take To The Top The Prepaid Plastic Card Business, Financial Times, Nov. 11, 1988, at 7.

Rowe, Au Revioir Le Cash?, Banking Technology, Jul.-Aug. 1991, at 46.

Okamoto and Ohta, Universal Electronic Cash, Cryptography Symposium (1991).

Primary Examiner: Willis; Davis L.
Assistant Examiner: Rashid; Peter J.
Attorney, Agent or Firm: Dowling; Thomas P., Bromberg; Laurence J.

Abstract

An improved monetary system using electronic media to exchange economic value securely and reliably. The invention provides a complete monetary system having electronic money that is interchangeable with conventional paper money comprising (1) issuing banks or financial institutions that are coupled to a money generator device for generating and issuing to subscribing customers electronic money including electronic currency backed by demand deposits, or electronic credit authorizations; (2) correspondent banks that accept and distribute the electronic money; (3) a plurality of transaction devices that are used by subscribers for storing electronic money, for performing money transactions with the on-line systems of the participating banks or for exchanging electronic money with other like transaction devices; (4) teller devices, associated with the issuing

and correspondent banks, for process handling and interfacing the transaction devices to the issuing and correspondent banks, and for interfacing between the issuing and correspondent banks themselves; (5) a security arrangement for maintaining the integrity of the system; and (6) reconciliation and clearing processes to monitor and balance the monetary system.

98 Claims, 69 Drawing Figures

[USPTO] [CNIDR]

United States Patent 5,420,405 ChasekMay 30, 1995

Secure, automated transaction system that supports an electronic currency operating in mixed debit & credit modes

Inventors: Chasek; Norman E. (24 Briar Brae Rd., Stamford, CT 06903). Appl. No.: 023,120 Filed: Feb. 26, 1993 Intl. Cl.:G06F 15/30 U.S. Cl.:235/379; 235/380; 902/ 4 Current U.S. Class:235/379 Field of Search:235/379, 380; 402/4

References Cited [Referenced By:]

U.S. Patent Documents 4,277,837Jul., 1981Stuckert 235/380 4,303,904Dec., 1981Chasek 235/384 4,454,414Jun., 1984Benton 235/380 4,614,861Sept., 1986Paulou et al. 235/380 4,630,201Dec., 1986White 902/ 4 4,689,478Aug., 1987Hale et al. 235/380 4,766,293Aug., 1988Boston 235/379 4,859,837Aug., 1989Halpern 235/380 4,906,828Mar., 1990Halpern 235/379 4,968,873Nov., 1990Dethloff et al. 235/379 5,175,416Dec., 1992Mansvelt et al. 235/380

Primary Examiner: Shepperd; John

Abstract

This invention describes a combination of methods and apparatus that creates electronic money for personal transactions which integrates the functions of cash, checks and credit cards with constant surveilance against fraud. This money can also serve as an international medium-of-exchange, and support automated sales tax collections and payment. This money's support system is comprised of personal terminals, vendor terminals, an electronic banking sub-system, and homebase terminals. Such a system, if widely used, would increase commercial and personal productivity, provide better security against fraud and counterfeiting, facilitate the automation of operations that involve currency, and sharply diminish the flood of paper that threatens to inundate the present system.

8 Claims, 14 Drawing Figures

Patch Work Agenda For The Off Site Meeting.

Koji Shimizu

System
- Front end & miller file system
- Batch system
- 5 local servers in the world
- Automated Account transfer system
- Pre registered account : ID card issuance
- Do not forget file maintenance risk
- Quality control

Sales

Collection methods & Distribution Channels
- Direct sales: Do we want to accept check ? Handling cost, Balance insufficiency
- Credit Cards :
- Automatic debit from tie-up banks account
- Coffee shops & drag stores: like lot
- Convenience stores

How to link POS system

Collection & merchant credit risk

Charges:
- Too much commission will create competitors opportunity
- Marketing gimmicks are more important : 100,000 vs. 10
- User charge
- Merchant charge

Bank

Key Function at the Genex Bank

We agreed with Mr. Nishi that the Bank will participate in the clearing part of this project
- Guarantee
- Settlement vehicle
- Legal vehicle
- Account

Gold Backed

Face value vs. Real value

Accounts

Interest

Settlement
- Account transfer
  - 1 week hold
- Cash outccapability
  - Pay through correspondent banks
  - How to keep balance or to avoid cash out

Data record

Sub Custodian
- Austria government
- US Federal Reserve

How to avoid Government Control
- Multinational legal control
  - Key - Where, Who,
- Foreign Exchange control laws
  - How to avoid Foreign Exchange license at 3rd worlds &
- Meaning of less than $1,000-
  - Money laundering laws

Existing banks cost structure makes loss from small settlement

Fake & Fraud
User side
Merchant side

Insurance
Cash protection service

Reserves
How to avoid over wrapped reserve requirement
Requirement :10 % should be hold as Gold & cash related collateral
Japan's regulation : Don't worry there is a loop whole
US & UK : It's depend on the product type
Other countries : ?

Who is the Genex Bank
Bland new baby
- Licensed - October '96
- Type of license Off-shore
- Regulator: Commonwealth of the Northern Mariana Island
- Established by the Genex Corporation Japan
- Share holding : the Genex Net Corporation

Lessons
World Cash project : Citicorp strategy do not prefer mass consumer
AMEX Optima : Data clash means bankruptcy
Mondex not to be a slave to Electric companies
Bills: Except few commercial banks, banks lost their pilferage
Credit cards : not all mighty in the world
Signature check: do you know how expensive to verify by human eye
Anglo- Saxon, continental, Latin and Asian law
Commonsense are different at countries and industries

Notes by Ara

Patch Work Agenda For The Off Site Meeting.

Koji Shimizu

System
- Front end & miller file system
- Batch system
- 5 local servers in the world
- Automated Account transfer system
- Pre registered account : ID card issuance
- Do not forget file maintenance risk
- Quality control

- Fast response must be guaranteed – 15 - 30 sec maximum.

data processing cost
data must be reliabile

Sales
Collection methods & Distribution Channels
- Direct sales: Do we want to accept check ? Handling cost, Balance insufficiency
- Credit Cards :
- Automatic debit from tie-up banks account
- Coffee shops & drag stores: like lot
- Convenience stores

Europe/Japan – many automatic debit services

How to link POS system
Collection & merchant credit risk — on-line is better

Charges:
- Too much commission will create competitors opportunity
- Marketing gimmicks are more important : 100,000 vs. 10
- User charge
- Merchant charge

Bank
Key Function at the Genex Bank
We agreed with Mr. Nishi that the Bank will participate in the clearing part of this project
- Guarantee
- Settlement vehicle
- Legal vehicle
- Account

Shimizu & Kunii will participate in the "HyperBank"

ISSUE TO DISCUSS

Gold Backed
Face value vs. Real value
Accounts
Interest
Settlement
- Account transfer
  - 1 week hold
- Cash outccapability
  - Pay through correspondent banks
  - How to keep balance or to avoid cash out

to create a float

○ Banks are credible trusted agents in Developing countries.
○ Geneo Bank has off shore bank which eliminates tax liabilities.

same day transfer at a cost

Data record

Sub Custodian
- Austria government
- US Federal Reserve

Try to avoid government control
Off shore banking

How to avoid Government Control
- Multinational legal control
  - Key - Where, Who,
- Foreign Exchange control laws
  - How to avoid Foreign Exchange license at 3rd worlds &
- Meaning of less than $1,000-
  - Money laundering laws

• Third world is very controlled. How can the control issue be resolved?

Important to avoid problems in most countries. $1000 is a good measure. USA: $3000 ⇒ long term record; $10,000 ⇒ IRS receives data.

Existing banks cost structure makes loss from small settlement

Fake & Fraud
User side
Merchant side
Bank side?

current investigate / pay threshold is $100 BUT multiple instances can trigger investigation.

Insurance
Cash protection service

controls Best done by external audit.

Reserves
How to avoid over wrapped reserve requirement
Requirement :10 % should be hold as Gold & cash related collateral
Japan's regulation : Don't worry there is a loop whole
US & UK : It's depend on the product type
Other countries : ?

> Transaction size: 1.5K - 3K. for BANKS.

Who is the Genex Bank
Bland new baby
- Licensed - October '96
- Type of license Off-shore
- Regulator: Commonwealth of the Northern Mariana Island
- Established by the Genex Corporation Japan
- Share holding : the Genex Net Corporation

Lessons
World Cash project : Citicorp strategy do not prefer mass consumer
AMEX Optima : Data clash means bankruptcy – DP errors caused inability to pay.
Mondex not to be a slave to Electric companies
Bills: Except few commercial banks, banks lost their pilferage
Credit cards : not all mighty in the world
Signature check: do you know how expensive to verify by human eye
Anglo- Saxon, continental, Latin and Asian law
Commonsense are different at countries and industries

Bankruptcy is very high in US, low in JAPAN.

$1000 is the break point

Suggestion: find appropriate break point for pay vs. investigation
find ways to minimize transaction costs

Big opportunity
Nationalization is important.

The Emerging World of Cyberpayments

Illustration by R. Jamison

Computing Japan is proud to have been selected as media sponsor (along with the *Yomiuri Shimbun*) of the upcoming CyberPayments@Tokyo conference.

by Wm. Auckerman

THE INTERNET IS A VAST new frontier of consumer-to-business and business-to-business commerce. For consumers, Internet-based shopping holds an attraction because of its breadth of coverage and ease of use. For corporations, Internet-based commerce represents an as-yet largely untapped medium for expanding and growing their business.

Putting up a Web site to promote and display products, and then luring online shoppers in to look around at the offerings, however, is only one dimension of conducting business via the Internet. To realize the true potential of electronic commerce (EC), an effective method of receiving payment for products sold or delivered through the Internet is a necessity. Developing and implementing effective and simple "cyberpayment" methods is a major focus of current Internet-related research.

Defining our terms

Terms like "electronic commerce," "e-cash," "cybermoney," and similar general and proprietary terms are tossed about casually in the press and in daily conversation, with little attention to just what [illegible] mean. Since there is as yet no standard terminology for th[illegible] of electronic-based payments, and the practitioners and [illegible] various financial sectors apply their own special (and often [illegible]) technical definitions to common terms, there is great confusion.

There basically are two types of electronic cash technologies: stored-value products (such as a JR Orange Card or an NTT telephone card) and access products (that process a transaction online, such as a bank ATM card). The concept of cyberpayments encompasses means of payment that use either (or a combination of both) of these technologies to effect fund transactions.

Primitive forms of electronic-based transaction systems have been around since last century (consider wiring "money" via telegraph). A more recent form of electronic transactions that we are all familiar with is credit cards and ATM (automated teller machine) cards, which contain an encoded magnetic stripe that enables access to a line of credit or bank account, commonly by transmitting the encoded information over a phone line or network for approval. Another common form is magnetic stripe cards that can store value (such as telephone cards). And being developed today are microchip-embedded "smart cards" that offer increased storage capacity, security, and applicability.

Internet-based payment systems take the concept of EC further, though, by facilitating financial transactions with other users. Since the personal computer that serves as the user's interface for Internet transactions can also store value, like the smart card it can act as both an access device and a stored-value device.

Looking to the future

Electronic systems like the Internet that facilitate the transfer of financial value promise to alter the means by which we conduct transactions. A common element of cyberpayment systems is that they are designed to provide transacting parties with immediate, convenient, and secure (even potentially anonymous) [illegible] transferring financial value. When fully implemented, [illegible] technologies will provide users worldwide with numerous be[illegible] legitimate commerce. Unfortunately, these technologies carry with them many of the theft/tampering, fraud, and privacy abuse dangers of traditional cash-based commerce, while creating new potentials for abuse as well. How, and to what extent, governments should and must become involved in regulating EC is an issue that is only now starting to surface.

Judging from media coverage of the potential of EC, and the interest being shown by corporations, banks, and government agencies, Japan obviously feels that it has once again fallen behind in a burgeoning global industry. The Japanese press is rife with stories about new EC-related service trials, corporate cooperative tie-ups, and government-backed R&D projects.

Since summer, the Ministry of Finance has been holding hearings to d[illegible] the extent to which Japan's foreign exchange and curre[illegible] need to be revised to cope with the new reality of EC. Se[illegible] ministries, including the Ministry of Posts and Telecommu[illegible] Ministry of International Trade and Industry, and Ministry of Justice, are looking closely at areas of the issue that fall (or that they hope will fall) under their purview.

Japan's banks, though, aren't quite sure where they stand on the issue of electronic cash. Many fear the effect that new technologies could have on their operations, as telecommunications firms and electronics manufacturers develop EC alternatives to traditional financial transactions. Yet they are reluctant to make the huge investments that would be needed for them to provide widespread electronic services themselves, for fear that the investment would be wasted if the concept does not catch on in Japan.

Will it catch on? As is the case with most new technologies, the media seem to be exaggerating where electronic cash is now, or will be in the next year, but grossly underestimating the revolution of realities in the next five years. If the past is a guide, consumer acceptance and use of cyberpayment and electronic banking technologies will grow slowly, but grow it will.

As with any market-based product/service, in the end only the most dependable, secure, and user-friendly EC strategies will succeed commercially. For corporations considering making an investment, with dozens of competing payment systems and standards in a constant state of evolution, it is imperative to know which will thrive and which will not. And that is where conferences like CyberPayments can prove invaluable.

CyberPayments@Tokyo

The inaugural CyberPayments conference, held in Dallas in June, was attended by 320 delegates from 10 countries. Following up on that success, the National Automated Clearing House Association (NACHA), which is responsible for the US's largest electronic payments system (over $10 trillion transmitted in 3 billion transactions annually), has scheduled a series of three similar events in the Asia-Pacific. CyberPayments@Tokyo, to be held on November 14 and 15, is the second of the series, coming after CyberPayments@Hong Kong (November 5 and 6) and followed by CyberPayments@Sydney (Nov. 26 and 27). The three Asia-Pacific CyberPayments conferences are being produced and managed by Financial & Business Media Associates, Ltd.

CyberPayments@Tokyo will be Japan's major international conference devoted to the latest developments in payments over the Internet. The sessions are geared toward helping those in attendance stay ahead of the competition by presenting advanced strategies, how-to tips, and forecasts from industry leaders, both Japanese and foreign.

As befits a conference devoted to developing technologies, a significant number of the sessions will feature online demonstrations of Internet payment systems and services. A brief look at some of the sessions and activities of corporations involved offers an instructive overview of important current issues.

Hiroshi Yuguchi, director of global multimedia for KDD, will cover emerging Internet technologies, services, and opportunities. Asked what new development it considered significant, KDD pointed to an agreement it has signed with Gemplus, a French company with top worldwide share of IC card production, to jointly market, distribute, and sell IC cards and related products in Asia. IC cards can be adapted to custom specifications and are thought to be less prone to misuse or forgery than traditional credit cards, bank ATM cards, and ID cards. KDD, which has already introduced prepaid IC cards for use in its international-call 011-IC Global Phones, is doing experimental research on Smart Pass, an Internet and LAN security method that employs an IC card authentication system. Because it carries out two-way authentication of both the client and the server, the Smart Pass authentication mechanism is especially suited to shopping via the Internet.

Mitsuo Yamaguchi, chief researcher with Hitachi Research Institute, will provide an international perspective of the trends and issues of cyberpayments, while Asahiko Isobe, senior executive managing director of Hitachi, will wrap up the conference by giving the closing remarks. Hitachi Research has provided *Computing Japan* with the summary of an intriguing study on payments for individual use that covers some of the hurdles to developing new payment systems and discusses the available strategies for banking institutions. Electronics/computer manufacturer Hitachi is an active participant in EC development in Japan and has been cooperating closely with Mondex, which is quickly emerging as the central player among the many electronic cash alternatives.

Nobuyuki Kinoshita, a director of FISC (The Center for Financial Industry Information Systems, an affiliate of the Ministry of Finance), will speak on the topic of electronic money in Japan. FISC, a nonprofit organization supported by some 1,000 corporate members, performs surveys and research on financial information systems and develops appropriate standards and guidelines. Based on the results of its last year's Study Group on Electronic payments, FISC is working to bring about the early realization of electronic payment systems in Japan.

Hisashi Yoshikawa, director of MITI's information, computer, and [illegible]nications policy division, will focus on the problems of p[illegible]k-based EC, including the establishment of transaction-a[illegible]aring responsibilities for losses, authentication, costs, and [illegible]on of privacy. Hideo Nakanishi, chairman of the steering committee of [illegible]ectronic Commerce Promotion Council of Japan) will br[illegible]pic by discussing the electronic commerce challenge. EC[illegible]hed through a MITI initiative in January 1996, serves as a[illegible] forum for examining the broad spectrum of issues relating to EC. An interim report of MITI's Study Group on Environmental Improvement for Electronic Commerce is available from the ECOM Website (http://www.ecom.or.jp/eng/ec_japan/ohp1.htm).

Watch our February issue for a follow-up of the issues covered (and uncovered) at the upcoming CyberPayments@Tokyo.◑

neo communication

neoについて

world wide web solutions:
2 byte programming
hosting • java™
creative engineering

http://www.neo.com
info@neo.com
ph: 415.512.9999
fx: 415.243.9901

ASIA'S MOST COMPREHENSIVE EXECUTIVE PROGRAMME DEVOTED TO PAYMENTS AND THE INTERNET

@Tokyo 1996

CYBER PAYMENTS

Westin Tokyo, Tokyo, Japan

14-15 November

Mitsuo Yamaguchi, Chief Researcher
Hitachi Research Institute, Japan

William Melton, Chief Executive Officer
CyberCash, Inc., USA

Dr. Marc Lassus, Founder and Chairman
Gemplus, France

Shuji Nakata, Senior Manager
Electronic Commerce
Business Promotion Division
NEC Corporation, Japan

Arthur Coleman, Product Line Manager
for E-Commerce, JavaSoft
Sun Microsystems USA

Bank of Japan
Mondex International, UK
The Center for Financial Industry Information Systems (FISC)
Electronic Commerce Promotion Council of Japan (ECOM)
MasterCard International
Dai Nippon Printing Company
KDD
Visa USA
Netscape Cimmunications
GTE

Attend CyberPayments@Tokyo for:

- ***Timely information*** on the size and scope of the potential market for commerce over the Internet
- ***Informative updates*** on the latest developments in transaction security
- ***Thoughtful analysis*** of the implications of cyber-banking for Japan's financial community
- ***Practical advice*** on creating a solid business strategy for Internet banking and commerce
- ***Real-time demonstrations*** of various payment systems to examine their appropriateness for everyday applications
- ***Valuable strategies*** for buying and selling on the Web
- ***Essential data*** on trends in Internet commerce utilising smart cards, credit cards, debit cards, electronic purse, digital cash and other payment systems
- ***Unique perspectives*** from the most respected authorities on Internet banking and commerce in the world

Sponsors:
HITACHI
Sun microsystems
DNP Dai Nippon Printing Co.,Ltd.
NEC

Co-Sponsored by:
KDD
MasterCard
GEMPLUS

Media Sponsor:
COMPUTING JAPAN

NACHA
safe • reliable • convenient

FBMA

http://www.nacha.org

OUTREACH

Press Releases

10-31-96 - Outreach Makes Proven Transaction Engine Available to Programmers

07-25-96 - Outreach Adds CyberCents Micropayment Service to Growing List of Internet Capabilities

03-22-96 - Outreach adds Counter Service to their growing line of Internet services.

01-22-96 - Outreach begins shipping its MallManager(tm) application

12-19-95 - Outreach rolls out STOMP transaction processing service

11-06-95 - Outreach pilots its secure, open Internet transaction processing service months ahead of any competition

09-05-95 - Outreach embarks on its effort to provide comprehensive Internet marketing and transaction processing services

---

Home Page|ShowCase | Rates and Services |About Us | CommerceLink(tm) | WebHelp | Feedback

---

Toll Free: 888-280-9999 International: 512-832-0132
The fine print

FOR IMMEDIATE RELEASE
Press Contact: Jack TaylorTate Austin Advertising & PR
(512) 476-7696
Jgtthree@aol.com

Outreach Adds CyberCents Micropayment Service to Growing List of Internet Capabilities

AUSTIN, TX, July 25, 1996--Outreach Communications Corporation has added the ability to handle small transactions to its growing line of electronic commerce technologies The new CyberCents micropayment system allows users to handle on-line transactions as small as one cent or less while maintaining profitability thanks to reduced transaction costs.

The system is the latest addition to Outreach's wide range of Internet transaction technologies, which already includes secure, real-time credit card sales transactions across the Internet that enable businesses to make sales on line.

"The addition of our new micropayment system adds more depth to the range of services that we can offer our clients," said Outreach president Julie Fergerson. "Few, if any, companies can match the broad array of Internet marketing and transaction processing software and services that we provide, and we intend to introduce even more electronic commerce technologies in the coming months."

The new CyberCents system is already in use at an Internet game site where players pay by the minute to play popular games against each other (http://www.cybercents.com/Arcade). Other anticipated applications include information sales, such as unit sales of electronic newspapers and the sale of photographs and other images.

As with Outreach's other transaction technologies, start-up costs for the CyberCents system are minimal. The company instead charges a small commission on each transaction. Outreach also offers businesses expecting high transaction volumes other payment options.

"No manager truly knows in advance how well their products will sell on the Internet" said Fergerson. "As a result, we offer pay-as-you-go transactions that take the risk out of the decision.''

Last year, the company pioneered Internet transaction technology by introducing its MallManager server software, which makes any web site a complete point of-sale. From one virtual store front to hundreds, customers visiting a web site powered by MallManager place products in virtual shopping carts by clicking on their purchases before paying with a credit card. MallManager handles all product options, such as available sizes and colors, as well as taxes and shipping costs. In addition, it provides full sales reporting.

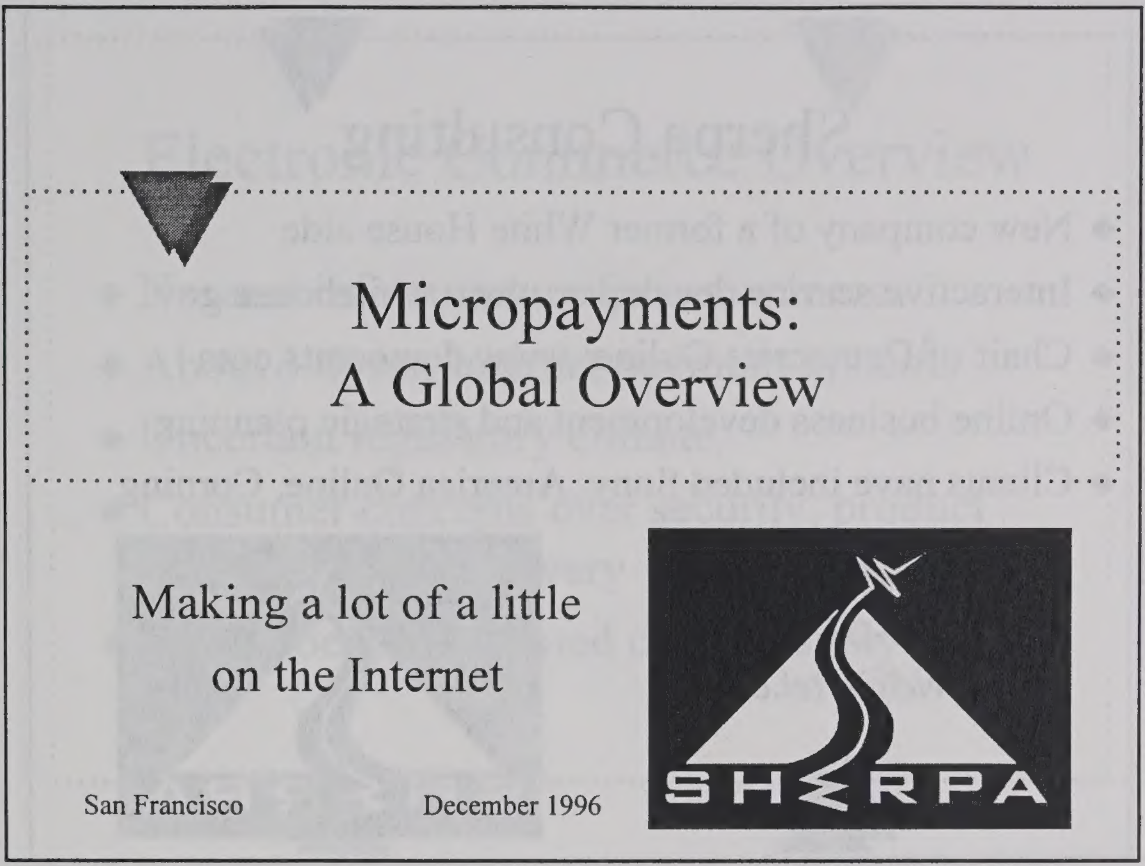
MallManager is made possible by another Outreach breakthrough called the Secure Transaction and Order Management Processor (STOMP). STOMP is a transaction end which enables any web site, secure or not, to conduct secures SSL-encrypted credit card sales. This proven technology is also a key component of the CyberCents micropayment system.

The Arcade site, CyberCents and MallManager all take advantage of real-time connectivity with the card networks. Because STOMP processes transactions in real-time, it enables real-time sales and automated distribution of electronic media, such as software, letting customers download their purchases only upon credit card approval.

The Austin, Texas-based company is a leading supplier of turnkey Internet marketing and transaction processing software and services. Outreach's clientele ranges from the Republican National Committee to the large software producer Data Tools to small business start-ups. The CyberCents micropayment system can be accessed at: http://www.cybercents.com. Information on Outreach's other transaction processing can be found on its home page at:http://www.outreach.com.

Home Page|ShowCase | Rates and Services |About Us | CommerceLink(tm) |
WebHelp | Feedback

Toll Free: 888-280-9999 International: 512-832-0132
The fine print



Micropayments:
A Global Overview

Making a lot of a little
on the Internet

San Francisco December 1996

SHERPA

The goal of this presentation is to provide you with a survey of the competition, to be used as a baseline in discussions of the viability of a new entrant in the field of micropayments.

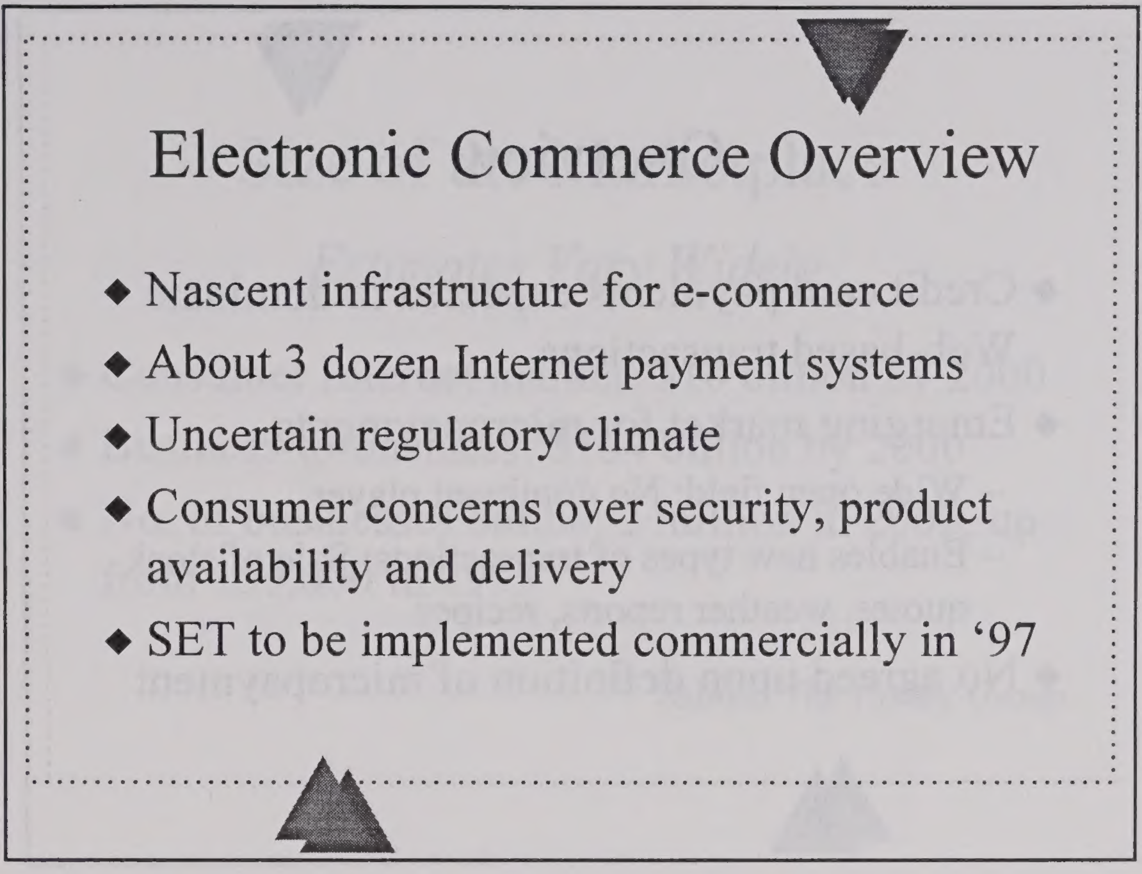
Sherpa Consulting

- New company of a former White House aide
- Interactive service developer: www.whitehouse.gov
- Chair of Democrats Online: www.democrats.com
- Online business development and strategic planning
- Clients have included Sony, America Online, Corning

www.sherpa.org
1-888-GUIDING
david@lytel.com

SHERPA

I was responsible for building and managing the White House Web site, which Wired has called "easily one of the best sites on the Net." When I left in April I began a consulting firm that provides online business development strategies, so please let me know if you're building or re-designing your Internet services and I can help. We also built and ran Democrats Online, which was an independent campaign committee in the 1996 election cycle. The site made it possible for people to interactively create their own personalized endorsement pages with materials drawn from the record of the Clinton Administration or the Democratic Party's accomplishments over the last 30 years. Visitors were also able to call a 900 number and record a testimonial that became part of their endorsement page. We also did something similar for America Online which became a feature on AOL called Lawn Signs in Cyberspace. There are copies of an interview I did on politics online that ran in Time Magazine's "Time Digital" edition up here if any of you would like one.

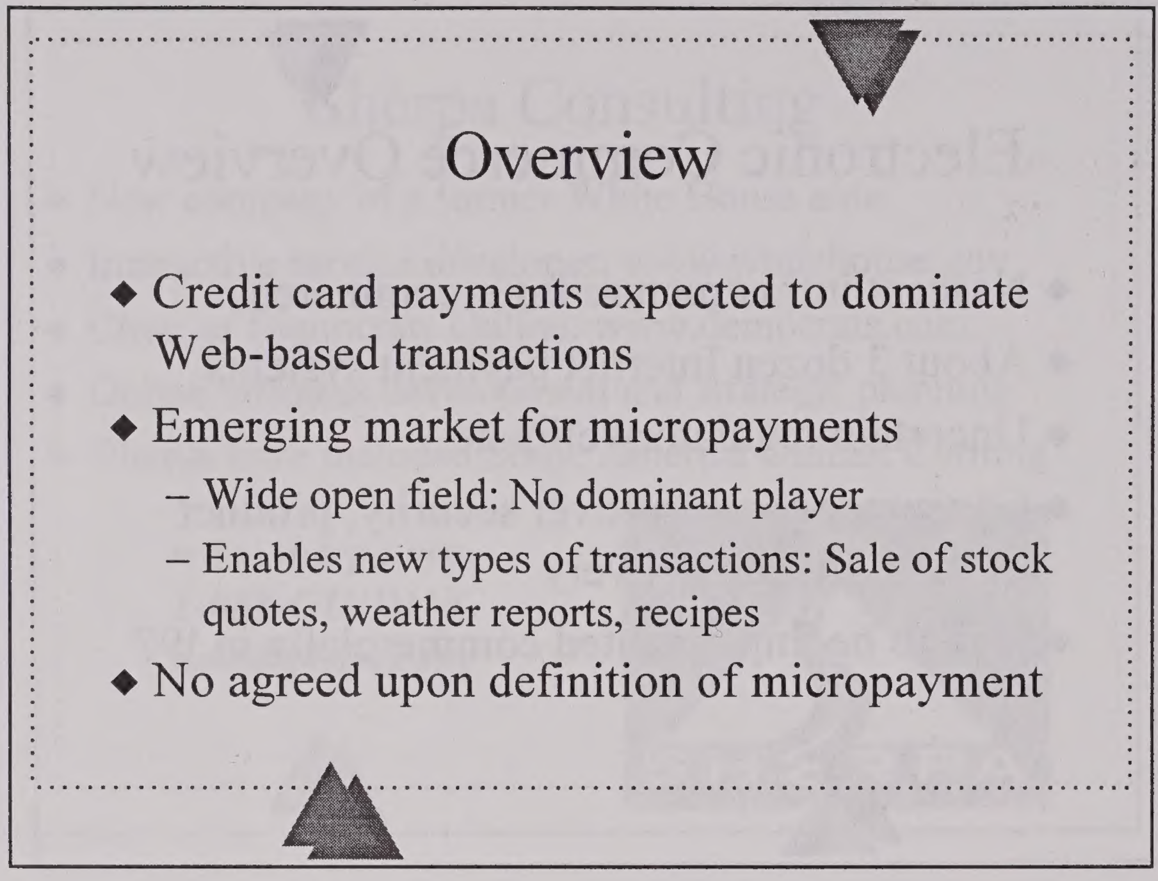


Electronic Commerce Overview

- ◆ Nascent infrastructure for e-commerce
- ◆ About 3 dozen Internet payment systems
- ◆ Uncertain regulatory climate
- ◆ Consumer concerns over security, product availability and delivery
- ◆ SET to be implemented commercially in '97

Significant competition for expensive transactions, which in the offline world have already been essentially captured by credit cards. Concerns over the exchange of money over the Internet are diminishing as well-known credit card companies, such as Visa and MasterCard, enter the online commerce market and provide users with the reassurance that their liability is limited and that they have a third party to go to as recourse for a failed transaction.

Even with the uncertaintly over regulation online commerce is growing rapidly and will accelerate with the implementation of the Secure Electronic Transaction (SET) standard next year.

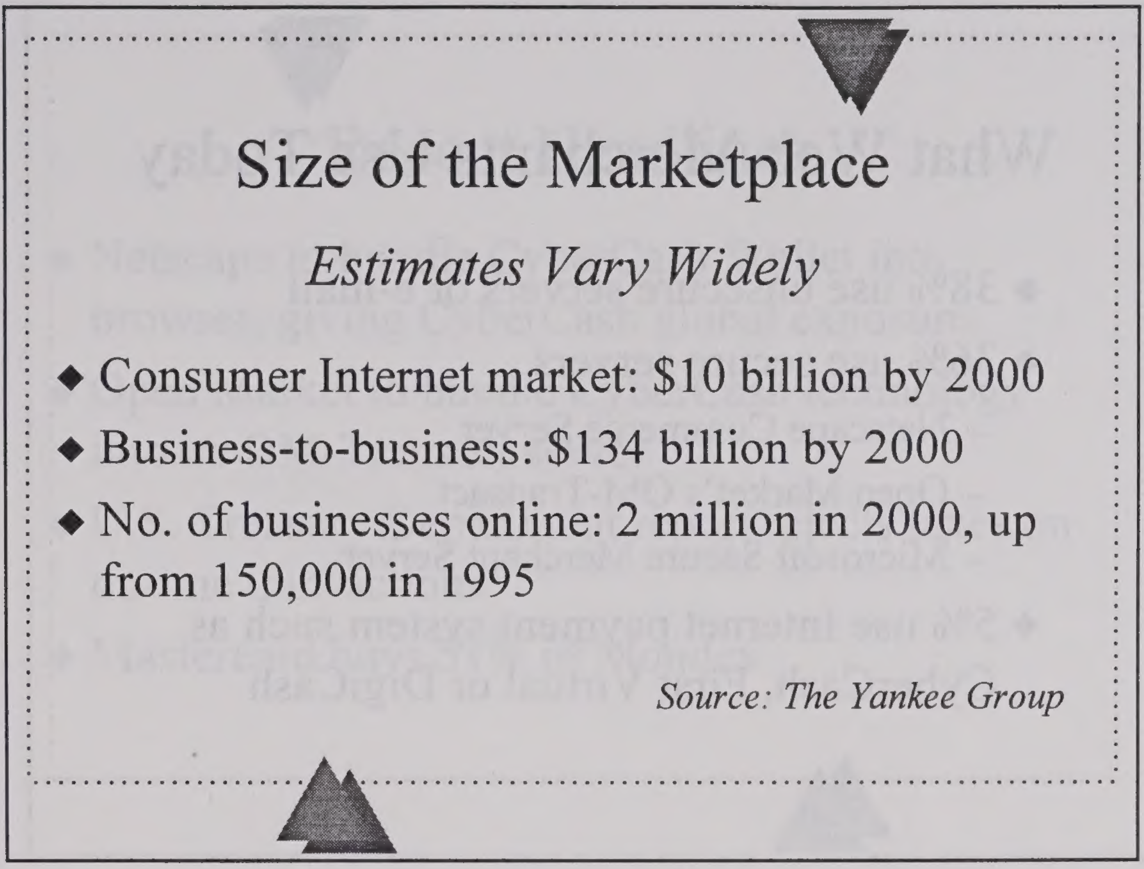


Overview

- ◆ Credit card payments expected to dominate Web-based transactions
- ◆ Emerging market for micropayments
  - – Wide open field: No dominant player
  - – Enables new types of transactions: Sale of stock quotes, weather reports, recipes
- ◆ No agreed upon definition of micropayment

However, the credit card infrastructure is too costly and the verification and settlement process too burdensome for small transactions. It is estimated that 85% of offline business transactions are for an amount less than $20 and that 97% of transactions less than $10 are done with cash.

In the micropayment arena--which we will define simply as transactions of less than 1 cent--there are no dominant players. The tantalizing idea behind micropayments is that they can be set at an amount that is low enough for the information consumer to be almost unaware of them, yet aggregated together efficiently enough for the information producer to make a living from this form of distribution.



Size of the Marketplace

Estimates Vary Widely

- ◆ Consumer Internet market: $10 billion by 2000
- ◆ Business-to-business: $134 billion by 2000
- ◆ No. of businesses online: 2 million in 2000, up from 150,000 in 1995

Source: The Yankee Group

The Yankee Group has a reasonable set of numbers for the market we're talking about here, but in truth no one really knows.

What Web Merchants Use Today

- 38% use unsecure servers or e-mail
- 36% use secure servers
  - Netscape Commerce Server
  - Open Market's OM-Transact
  - Microsoft Secure Merchant Server
- 5% use Internet payment system such as CyberCash, First Virtual or DigiCash

Most transactions on the Internet today are unprotected, although the use of secure servers is growing steadily. Fewer than one transaction in twenty uses one of the new Internet payment systems.

Recent Headlines

- Netscape to bundle CyberCash Wallet into browser, giving CyberCash global exposure
- Open Market to bundle CyberCash technology into its OM-Transact server
- U.S. Treasury Department report rejects taxes on on-line transactions
- Mastercard buys 51% of Mondex

More Recent Headlines

- W3 and CommerceNet complete JEPI standard
- CommerceNet and EFF partner to implement eTRUST
  - Global initiative for establishing consumer trust and confidence in electronic transactions
  - To establish an accreditation program for companies conducting transactions

TheWorld Wide Web Consortium's Joint Electronic Payments Initiative (JEPI) is a universal payment platform that will allow merchants and consumers to transact business over the Internet using many different forms of payment. In other words, JEPI allows for the interoperability of payment systems. That's the significance. Some industry observers contend that the W3 lacks clout because it is primarily made up of academics. It is unclear whether JEPI will become widely adopted.

eTRUST will create an integrated logo system that eTRUST hopes consumers will associate with trust and confidence in electronic transactions.

eTRUST involves an accreditation program for companies seeking to conduct online transactions. The Commercial Guidelines and Auditing Standards Committee of eTRUST will develop guidelines for the accreditation program. Companies on the committee include Coopers & Lybrand, Firefly Network, KPMG and Test Drive.

Macropayments: Four Major Players

Each based upon a different technology

- CyberCash: Credit cards and electronic coins
- First Virtual: E-mail tied to credit card
- DigiCash: Electronic money (token)
- Mondex: Stored-value card

Other Influential Players

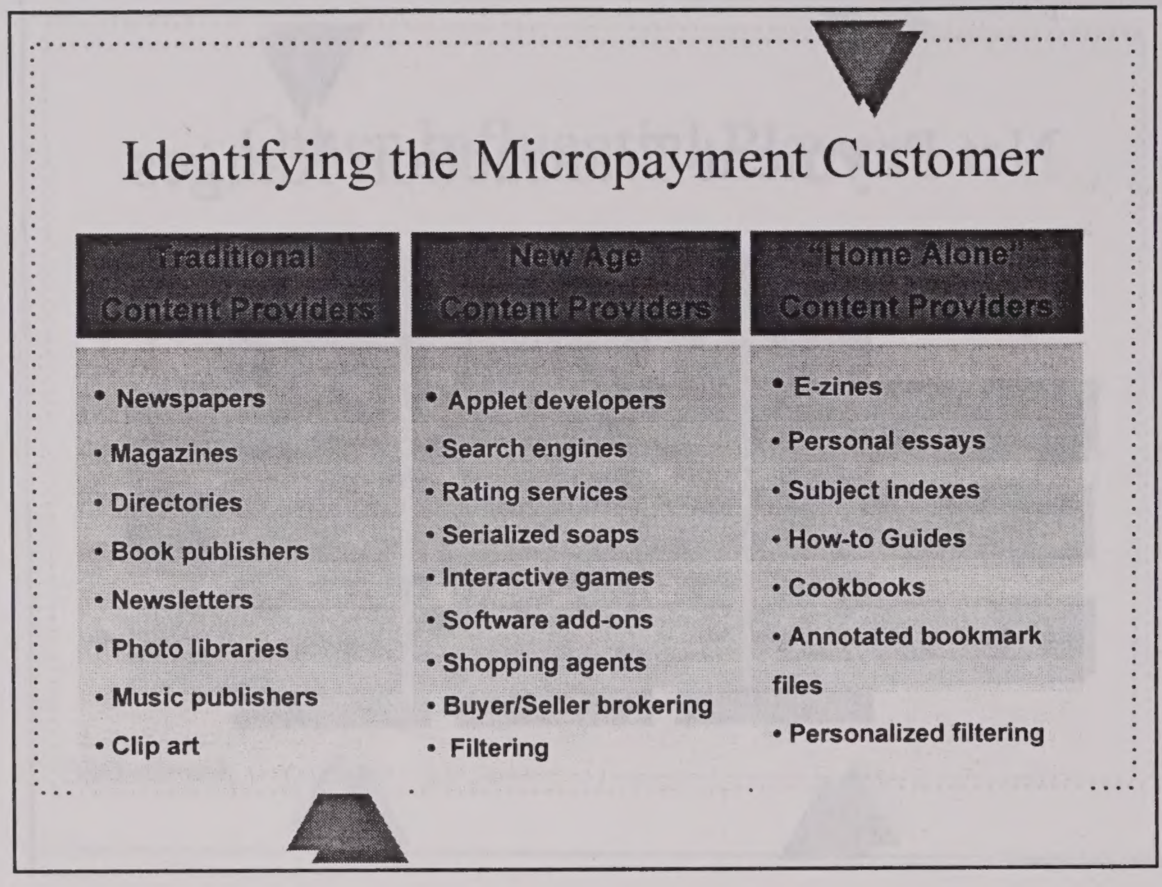
- Netscape Communications: Commerce Server
- VISA/Mastercard SET standard
  - Widely supported by payment systems, such as CyberCash and Netscape
  - Not meant for micropayments: Use of public-key escrow raises transaction costs
- Microsoft: Secure Merchant Server

SET is not a payment system but a credit card protocol developed by credit card rivals Visa and MasterCard that is expected to be implemented commercially in 1997. Though the impact of SET is unclear, it is likely to make credit card transactions more widespread as a growing number of payment systems companies integrate their protocol into their payment systems.

Net Payment Transaction Ranges

Payment	Minimum Transaction Value	Typical Transaction Value	Maximum Transaction Value
Macro	$5.00	$50.00	$1000.00
Mini	$0.10	$1.00	$10.00
Micro	$0.001	$0.01	$1.00

Source: DEC

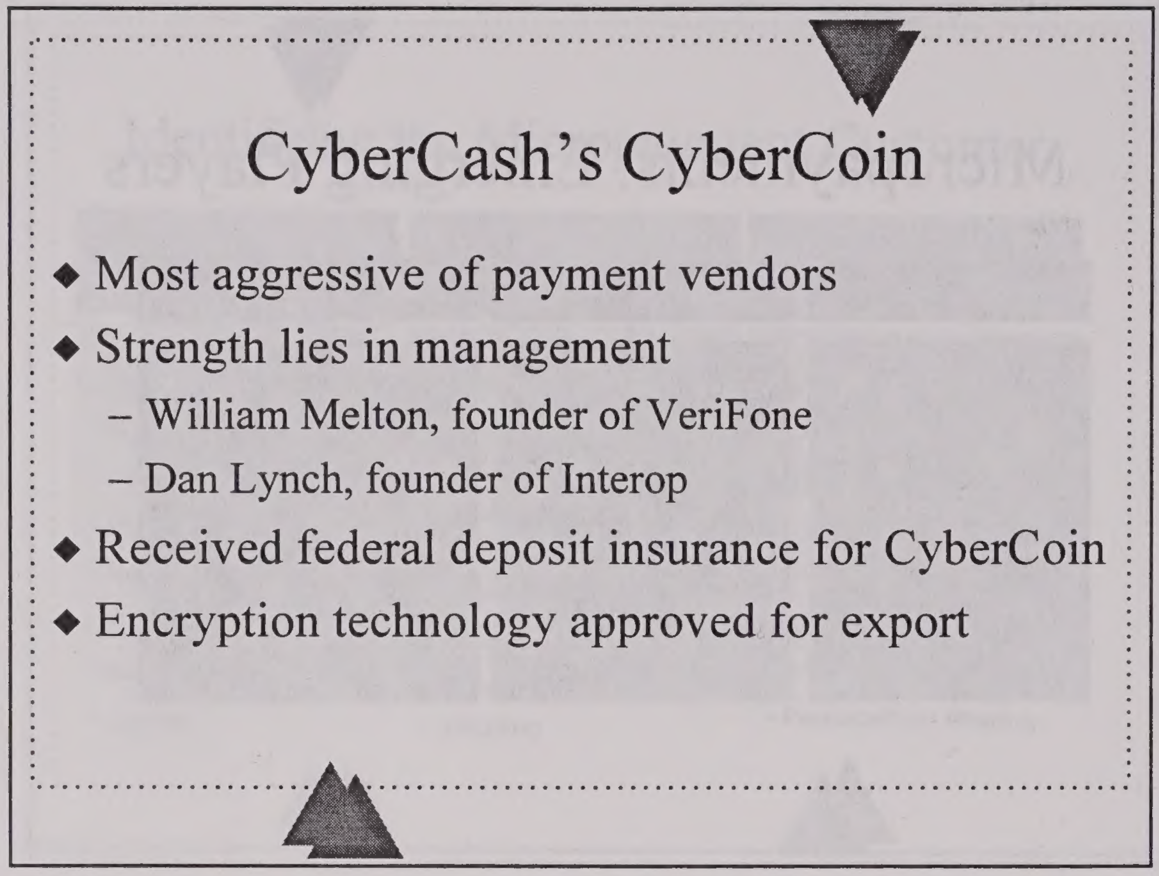


Identifying the Micropayment Customer

Traditional Content Providers	New Age Content Providers	"Home Alone" Content Providers
Newspapers	Applet developers	E-zines
Magazines	Search engines	Personal essays
Directories	Rating services	Subject indexes
Book publishers	Serialized soaps	How-to Guides
Newsletters	Interactive games	Cookbooks
Photo libraries	Software add-ons	Annotated bookmark files
Music publishers	Shopping agents	Personalized filtering
Clip art	Buyer/Seller brokering	
	Filtering	

Micropayments: Emerging Players

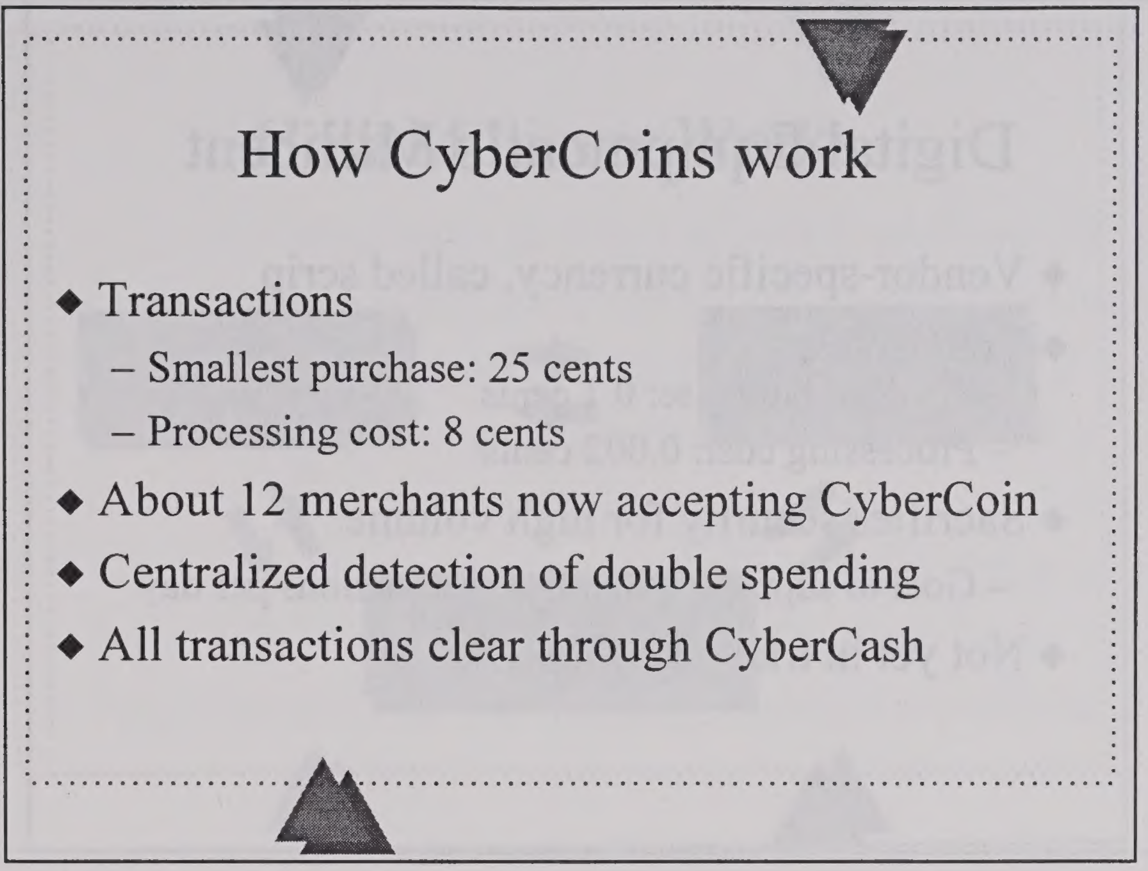
Developer	Project Name	Transaction Type
CyberCash	CyberCoin	[illegible]
Digital Equipment	Millicent	Scrip
DigiCash	ecash	Digital cash
Clickshare	Clickshare	[illegible]
[illegible]	[illegible]	[illegible]
[illegible]	[illegible]	[illegible]



CyberCash's CyberCoin

- Most aggressive of payment vendors
- Strength lies in management
  - William Melton, founder of VeriFone
  - Dan Lynch, founder of Interop
- Received federal deposit insurance for CyberCoin
- Encryption technology approved for export

CyberCash has established itself as the dominant player, so far. It has built a network of partnerships and alliances with software, hardware, financial and Internet access companies. Last year, CyberCash received federal deposit insurance for CyberCoins, making it the only non-bank entity that has FDIC insurance. And last May the Commerce Department gave CyberCash approval for an export license for its CyberCash Wallet using 1024 bit keys that scramble data. It also has an electronic check service under development. Founded in 1994, CyberCash so far has lost about $20M and is expected to lose another $20M. Analysts expect it to break even in 1997.



How CyberCoins work

- Transactions
  - Smallest purchase: 25 cents
  - Processing cost: 8 cents
- About 12 merchants now accepting CyberCoin
- Centralized detection of double spending
- All transactions clear through CyberCash

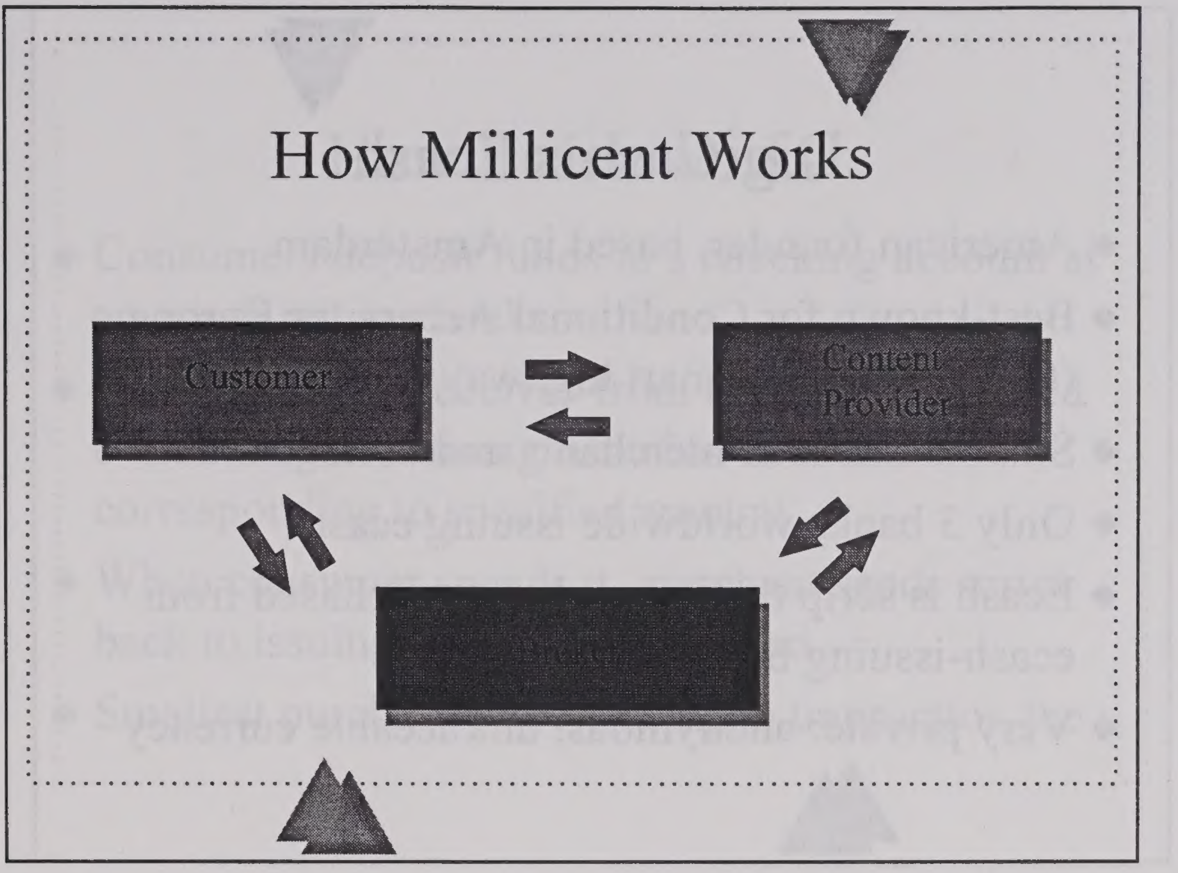
CyberCoins are not token-based, a hard-to-forge series of serial numbers. Instead they are real money that works like a debit card. Users transfer money from their personal checking account to an FDIC-insured trust account via their CyberCash wallet. The funds are then parked in the trust account until a purchase is made, at which time the funds are withdrawn from the trust account and transferred directly to a merchant's bank account.

Currently, CyberCoin transactions can only occur with U.S. dollars and with merchants that have a U.S. bank account. CyberCash is working to expand internationally.

The weakness of this system is its centralization. All transaction must pass through CyberCash, which makes its money as a percentage of each transaction.

Digital Equipment's Millicent

- Vendor-specific currency, called scrip
- Transactions
  - Smallest purchase: 0.1 cents
  - Processing cost: 0.002 cents
- Sacrifices security for high volume
  - Goal to support 2.5 million transactions per day
- Not yet in trial development



How Millicent Works

Customers purchase vendor-specific script from their broker which is used for purchase. The script consists of the vendor's name, value, expiration date and a hash which is a proof of validity.

Some of its advantages include the anonymity of the customer, forgery is hard and double-spending easy to detect, and script cannot be stolen because the payment is cryptographically tied to the request. The broker serves the role of the aggregator who handles billing the customer, making it easy to be a vendor.

DEC anticipates that large established financial institutions such as credit card companies will become brokers.

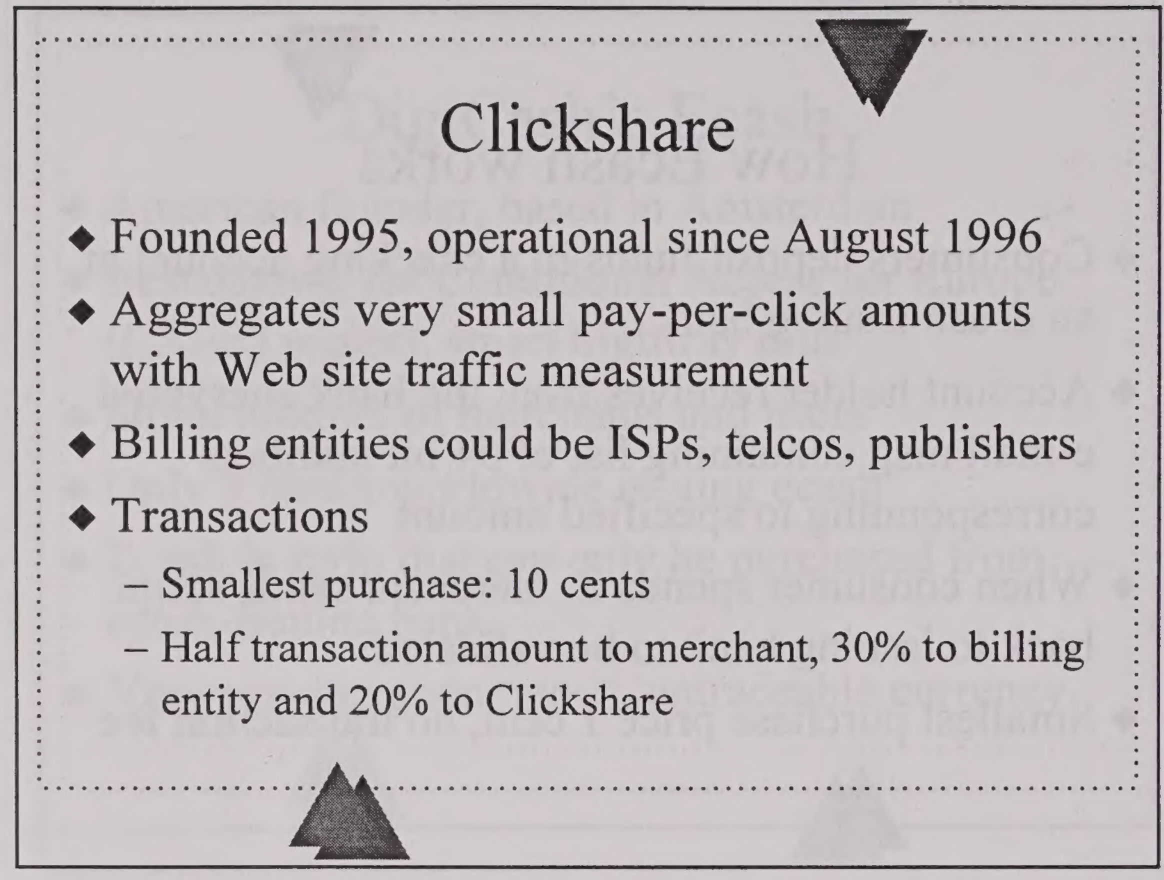
DigiCash's Ecash

- American founder, based in Amsterdam
- Best-known for Conditional Access for Europe (CAFE) project, smart highway tolls
- Small number of merchants and users
- Only 3 banks worldwide issuing ecash
- Ecash is scrip that can only be purchased from ecash-issuing bank
- Very private: anonymous, untraceable currency

DigiCash has about 50 merchants and fewer than 5,000 users worldwide. DigiCash is cumbersome to use because of the number of steps required by the user to set up an account. Users have to download the application form, fill it out and mail it back through the U.S. Postal Service, enclosing a deposit for establishing their account. Users then have to download and install the DigiCash software.

How Ecash works

- Consumers deposit funds in a checking account at an ecash issuing bank
- Account holder receives from the bank encrypted e-mail msg containing list of 64 bit numbers corresponding to specified amount
- When consumer spends it, merchant sends script back to issuing bank to be validated
- Smallest purchase price 1 cent, no transaction fee



Clickshare

- Founded 1995, operational since August 1996
- Aggregates very small pay-per-click amounts with Web site traffic measurement
- Billing entities could be ISPs, telcos, publishers
- Transactions
  - Smallest purchase: 10 cents
  - Half transaction amount to merchant, 30% to billing entity and 20% to Clickshare

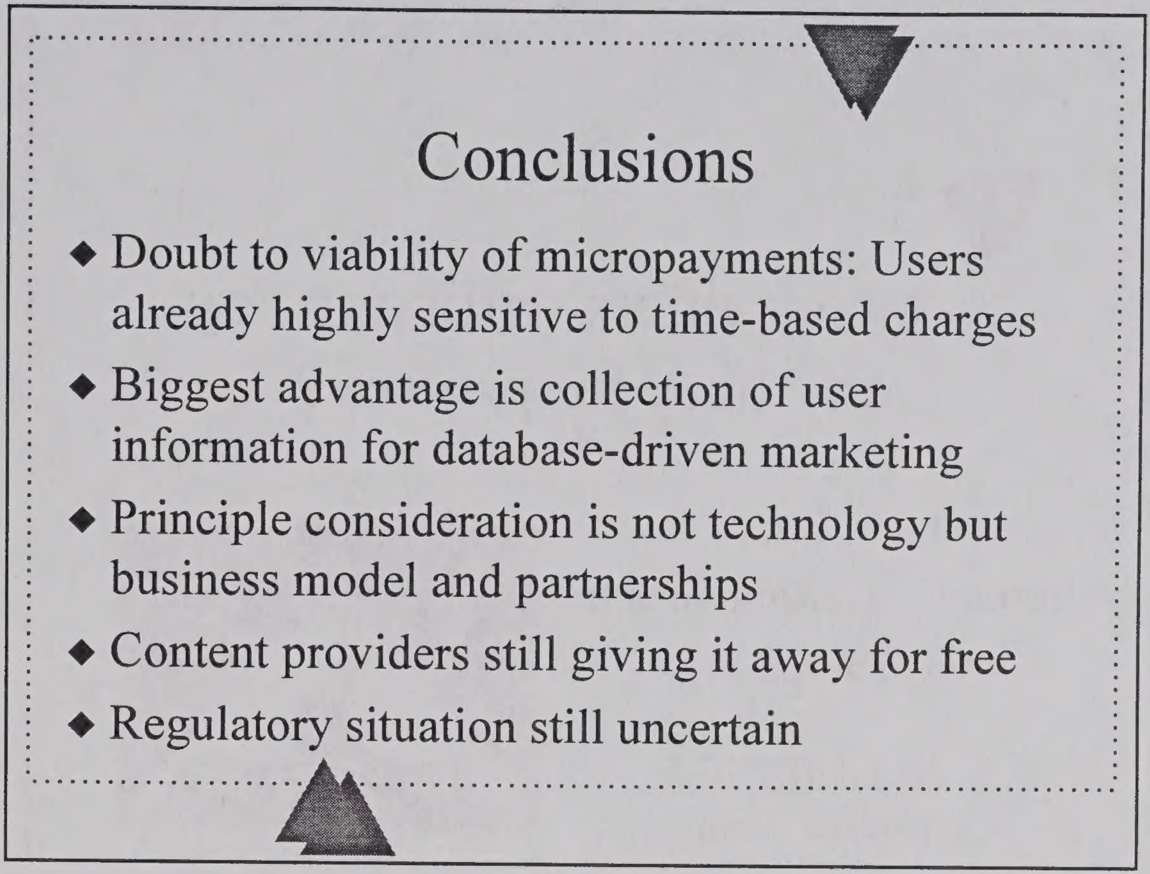
Users enroll with a billing entity, which could be an Internet service provider, telco, or content provider. When they visit a Clickshare-enabled site they are confirmed as a Clickshare user. The site records the user's consumption and delivers that information to the billing entity. Users receive a periodic bill on their credit card. The service also provides aggregated demographic information for non-charged pages, making it popular among advertisers and marketers.

Lurking Players with Potential

- Mondex: Proprietary, hardware-based system, highly secure but expensive to implement
- IBM: Cryptolope has additional benefit of inhibiting duplication
- VeriFone: Ready-made distribution network based upon installed base of 5M credit card terminals in 100 countries

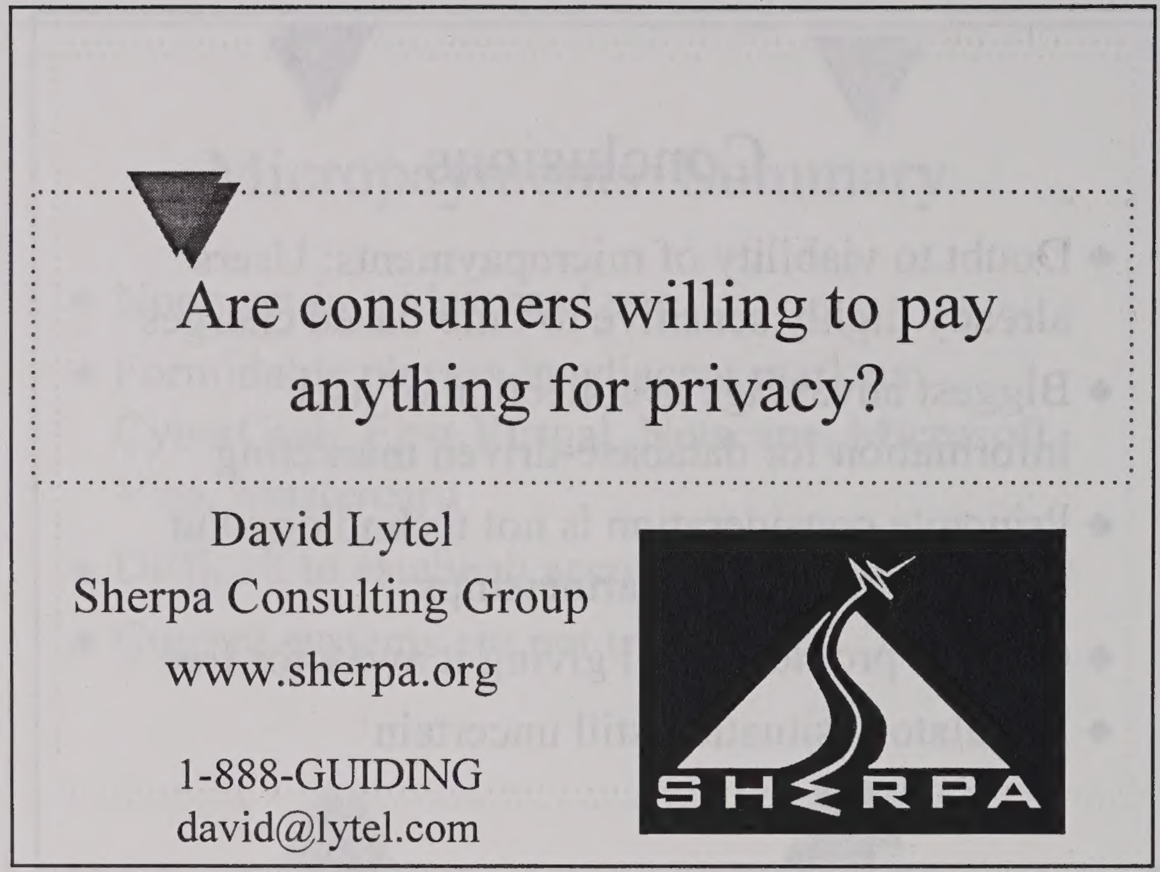
Micropayments: Summary

- None yet in widespread use
- Formidable players in adjacent markets: CyberCash, First Virtual, Netscape, Microsoft, Visa, Mastercard
- Difficult to establish accounts, not much to buy
- Current systems are not truly global



Conclusions

- Doubt to viability of micropayments: Users already highly sensitive to time-based charges
- Biggest advantage is collection of user information for database-driven marketing
- Principle consideration is not technology but business model and partnerships
- Content providers still giving it away for free
- Regulatory situation still uncertain



Are consumers willing to pay anything for privacy?

David Lytel
Sherpa Consulting Group
www.sherpa.org

1-888-GUIDING
david@lytel.com

One of the mistakes I think you won't see made again is the tremendous over-investment of resources in real time interaction. Schedulers and press secretaries have already learned that the senator or congressman can reach more people on any licensed radio station in the country than they can in even the largest online audience on AOL or CompuServe, and the number on a Web site is almost always smaller than the number of people in this room right now. That is not true about e-mail but it certainly is true of unscheduled, episodic real time interaction which is a real dud.

Remember that communicating online isn't magic. If politics is stale and boring offline then it isn't going to be much more interesting on the Internet.

So that's where I think we are today. If a journey of a thousand miles starts with a single step then I think we're still tying our shoes.

Notes from HyperCoin 2

Goals for the system (Nishi)

The idea is to make a electronic payment system which supports TransPublishing (pimarily media and links) and the supporting concepts of transcopyright and transclusion. The goal of the payment scheme is to satisfy that portion of the space where the granularity of transactions is tiny (under $ 0.05). The focus on the system should be global (that is, international).

Concerns and goals regarding the transpublishing system

In this document, system refers to a electronic payment scheme for royalties.

1. The financial system is targeted to the transmission of multimedia documents but must also comprehend other kinds of transactions including simple transfers of funds.
2. Currency is based on gold.
3. TransPublishing (explained). What is the extent of the rights conferred? Can a copy be passed along to other individuals once purchased?
4. Design for strength. Hard to spoof. Robust under attack. Extent to be decided. Formal validation where possible, especially for protocols.
5. Caching issue. Problem is with proxy caches, which allow multiple references to a document to be satisfied outside the payment mechanism. Queries are generally not cached. Site licenses are a potential issue. (See Lisa Sanger's paper, distributed.) Caching has licensing implications. Library definition and function. How to deal with copies, loans, and transfers of information.
6. Privacy and Anonymity. Transactions should be private and confidential. Statistics gathering should be difficult when it reveals information about individuals.
7. There is a formal notion of a transaction. Negotiation may be supported. Transactions should be atomic and conservative. There needs to be full recourse. Incomplete transactions should not be charged. A formal transaction model is needed particularly for verification and validation of protocols.
8. The system should accommodate small publishers. The publishing model is a peer model rather than a client server model. Every network user is a potential publisher. Myravold's law: most documents have a very few readers; few documents have many readers.
9. Risk thresholds could be constrained. There need be not mechanism to protect the privacy of those who are attempting to compromise the system. The system should not be at risk to large losses. For example, key compromise should have limited impact. We need blame manager and to identify those components which introduce potential risks. Risk exposure should be identified. Subversive collusion risks need to be identified.
10. There needs to be a scam analysis.
11. There needs to be intrusion and exception situation monitoring mechanisms built into the system.
12. What authentication can be provided? Which components of the system should be within the authentication model? What level of authentication do we want to offer? Authentication should not require tracing previous history.
13. System should be easy to use, difficult to misuse. Publication should be easy and fit into current paradigms of document production.
14. Support very small payments.. The granularity scale should be in nano-NUCs. The system should be cost effective in terms of processor resources, network resources, and the like. This may include use of transaction aggregation and other optimizations. Server placement may represent a significant optimization.
15. Decentralized and distributed control. Open interoperable system. Rogue transactions should not break the system or place its utilization at risk. Insofar a possible, denial of service attacks should be prevents.

16. The system needs to be able to provide reports of payment analysis, royalties, transport, storage, and other services. There are a number of different methods for managing royalty payments. Support for publisher, agent, and author needs need to be supported.
17. There needs to be a standard procedure for establishing the value of the gold used as a base for the valuation of information commodities.
18. The system needs to be neutral party participation without a significant observable profit motive to build trust. There is clear benefit to the product/service by involving a neutral party.
19. Transactions within the system can be trusted; the system is a guarantor for all transactions. A ombudsman must exist to correct problems.
20. Appropriate auditing, assurance, and reinsurance mechanisms need to be available. This is complicated by the international nature of the system. There needs to be neutral third party which verifies the state of the HyperMint system and gold based currency. There needs to be a standard for what pricing is used for gold, for example, 9am setting of gold price in London.
21. Minimize potential for guilty brokering. Guilty brokering would involve the HyperMint in a transaction which I plausibly illegal. The system is arms length from actual delivery of content.
22. The system needs to be able to function in all national environments and conform to all regulations, insofar as possible. Some nations may be declared inhospitable. The system cannot be predicated on changing existing governmental structures and policies.
23. Authors (rights holders) may specify a variety of policies for the distribution of their documents. The rightsholder can specify the details of the license under which the document is transmitted.
24. There should be a variety of standard pricing models available to publishers. The details of the pricing and distribution models must be independent of any component of the system.
25. The system may have a number of different clientsa . The Xanadu network service is one client of the system. Xanadu provides a storage, distribution, and publishing service. Xanadu receives all payments for documents under it's control, and disbuses royalties to the rights holders and take a commission for this service. The main business is storage rental and delivery.

Xanadu Role

Xanadu provides a network service. Xanadu is meant to be a licensed (franchised) system of servers under a common publication policy and under a shared distribution policy. It provides:

- Publish by name
- Guaranteed long term availability
- Support for OSMIC
- Consistent pricing term

Relationships and Friends (for future discussion)

- Banks
- Money/Gold Exchanges
- National Tax Systems
- Conventional Publishers
- Individual Rightsholders
- Libraries
- Educational Institutions

Open Controversial Issues

- Privacy Concerns
- What information can be gathered?

Issues List for Demo and Prototype

Xanadu

1. Need name resolution to map URN to URLs
2. Need to support media types beyond images
3. Does not use existing http information to advantage.
4. Need to support multiple distributed name servers.
5. Need to support negotiation model.
6. Database with pricing models.
7. Receipt to user

Interface

1. Need to provide stable local storage beyond cookies.
2. User authentication via http user authentication facility
3. No accounting authentication
4. Failing: complete customer trust.
5. Need risk analysis
6. Need support for privacy, prevention of eavesdropping, etc.
7. Need audit support.
8. Need user query interface (current account state in real currency)
9. Need specified source of financial information (e.g., price of gold)
10. Whatever that's done it should not break current practice and should be browse independent.
11. No insurance/assurance.
12. No neutral custodian

Mint System

1. Need formal specification of the syntax and semantics of all user transactions
2. Need to support multiple distributed mints.
3. Need to support multiple distributed transactions.
4. Need to support aggregation in all transactions.
5. Choice of user authentication model
6. Needs customer identifier (for account resolution)
7. System should be more peer-to-peer than client-server.
8. Limits on transactions, conditional transactions, etc.
9. Need internationalization support
10. All failures resolve in bank's favor. Some are not detected.
11. Fraud tripwires
12. Need to support range specifications

Demo Review

Current review has new name service.

Current embedded system:

<IMG 1 SRC = http://www.hypermint.net/buy?http:/www.snerd.com/images/ted.gif;50>

Current system:
) Browser -> HyperMint IMG request
2) Broser holds balance in a netscape cookie
3) Hypermint determines validity of the cookie; error message for invalidity
4) Mint does actual accounting
5) Browser sends redirect message to the browser indicating that the completion of the transaction.
redirect:http:www.snerd.com/images/ted.gif;value-string-50XAU

Use URN tag in cgi, implement via redirect.

urn://xanadu.com/OPAQUE_STRING

equiv

http://registry/xu/OPAQUE_STRING with appropriate cgi script
urn://xv/OPAQUE_STRING

which does the lookup and gets the appropriate URL.

-

Propose revision:

Fundamental Properties Checklist

Value
Exchange mechanism
Storable and retrievable
Tamper resistant
Authentication
Nonrefutable (receipt/proof of payment)
Multiple policies (negation)
Accessible to users
Reliable
Private
Protected, fraud resistant, protected against eavesdroppers
Accounting (divisibility, receipts)
Convertible (to/from cash, exchange)
Limits (if any in terms of time, amounts, conditional payment, post-dating)
Traceability
Anonymity

Other issues

Obligation
Escheatment
Universality
Regulation
Fees, float, redemption
Taxes
Accounting

To appear in *Agoric Systems: Market Based Computation*, edited by Wm. Tulloh, Mark S. Miller and Don Lavoie. This may be found by ftp at netcom.com:pub/joule/DSR1.ps.gz, DSR1.rtf.gz or DSR1.txt

> The art of progress is to preserve order amid change
> and to preserve change amid order.
>
> *Alfred North Whitehead,*
> Science and the Modern World

The Digital Silk Road

Norman Hardy
norm@netcom.com
Ph: 415 851 2582

Eric Dean Tribble
tribble@netcom.com

Introduction:

Existing and proposed mechanisms for digital money all require large overhead to transfer money between parties. This overhead makes them unsuitable for extremely low cost activities such as delivering and routing packets. We propose a money system with extremely low transaction cost built into the communication protocols. The money introduced by this system is much more like coins than like bank accounts; it supports only small transactions, requires limited trust among the participants, and requires no central bank. With this as a foundation, we then describe elements of an open system that fully supports network resource management, routing, interconnection with the Internet, and other information services, across trust boundaries with competing providers for all services.

The protocol provides incentives for communication and information providers that avoids policy entanglements typical of subsidized systems—cash-and-carry for cyberspace.

We design incentives based on positive reputations and profits in real local currencies. We take inspiration from tales of the silk road by which silk was brought from China to Europe by a series of traders who knew neither.

We use the term *site* here to describe some assemblage of hardware and software operated by an individual or enterprise. We do not distinguish the entities within a site in this paper and we assume that their interests are entirely aligned.

The Money Field = The Packet's Worth

The basic idea is an inter site link level protocol that includes a money field in some packets. This field is not ciphered and is never negative. It is 32 bits long and its value is the *worth* of the packet. It is denominated in units of one thousandth of a cent or less. As such packets pass over an interface between two sites, X and Y, an up-down *accumulator* that is conceptually between X and Y tracks the net money flow. The accumulator is probably implemented at both sites. Conceptually the respective operators of X and Y periodically read the accumulator and pass real money according to the accumulator's value, then reset the accumulator. More realistically this is accomplished by conventional EFT.

Communication is achieved by a network of sites interoperating with this protocol. Some sites specialize in moving data and others in switching while many do both. We call movers and switchers *carriers* here. One simple business for a site is forwarding datagrams. Such sites make a living by accepting packets at an interface, moving them to another interface, deducting a small amount from the packet's worth as toll and then delivering them across the new interface. Such packets include *steering information* in the packet—an indication for each site of the next interface the packet is to be delivered to, with that indication removed from the packet as it is forwarded. This business ensures a net income for such a site at its collective interfaces. Delivery of datagrams requires no tables or records kept by each site. When you send a datagram thru the network you include more money than will likely be required. The excess will arrive with the packet at the destination and can be returned with an acknowledgment. The amount returned is an immediate measure of the cost of the service.

At international interfaces a multiplication of packet wealth converts currency. Any site can set a money field any way it wants but successful sites are constrained as follows. Liabilities (obligations) arise as packets with worth are exported across an

interface. The accumulator there records its obligation to the next site operator. The site will thus be careful not to set it higher than the situation warrants. If sets it too low it will soon damage its own reputation and lose business thereby. Prices are set by normal market mechanisms of supply and demand. They may be set for periods of time and advertised to aid planning, or they may be more volatile. The market will decide this too. Since you pay up front you incur no obligation beyond what you pay and you lose at most what you pay.

Link Efficiency

We assume that packets are grouped into blocks and delivered across the physical interface with error control only for the blocks. This reduces the expense of small packets for they need not individually carry error control and acknowledgment information. This detail is necessary for realistic cost estimation. More suggestions on link strategy are in another paper.

Circuits

While datagrams and their incentives are easy to understand we advocate a more complex and efficient mechanism to move data involving persistent *circuits*. Circuits avoid the cost of steering information in each packet. We mark circuit packets across interfaces as belonging to a particular *channel* of that interface and we identify *agents* within sites that act for a particular circuit at that site. Thus a circuit is a data path from origin to destination consisting of channels across interfaces and agents at sites that work for the circuit. Each agent and channel serves just one circuit. The agents may be thought to *own* the interface channels for their circuits. As a circuit packet crosses an interface it is delivered to the agent that owns the channel of the packet. After possible buffering the agent delivers the packet thru its channel on the next interface. Most circuits are symmetric and can carry data in either direction.

Some circuits may provide flow-control or backpressure. This is done thru signals flowing between agents thru channels against the data flow indicating when more data can be accepted. Backpressure may arise at the destination or some intermediate point in the circuit. Circuits that don't do backpressure discard data when unable to deliver it. We will mention numerous other potential circuit qualities in a later paper.

An agent holds some money to rent buffers and pay for transmission across interfaces. Occasional packets thru the circuit carry money to replenish the agent's money.

Building circuits

Building a circuit occurs with a needle moving thru the network with steering information like a datagram but leaving a path of channels and agents behind it. To delete a circuit a destruct signal travels along the circuit that deletes the agents and frees the channels. By convention a zero in the needle as interface number signifies end-of-circuit and the current site is the target of the circuit. Messages thru the circuit are interpreted locally.

Note that it is possible to send a datagram using the circuit primitives: Send the needle, the data, then the destruct signal.

Routing Services, or Scouts and Guides

A scout program situated at some site in the network can explore available routes. It does this by launching circuits in all directions. As each site is reached the scout asks the site its name. The response is returned over the circuit. When the scout has not seen the name before, it adds the site to its map. The scout asks the site what it does for a living. When a site responds that it moves or switches data and the scout has seen the name before it adds a link to its map. Other responses are entered into an information service data base. At new sites the scout asks how many interfaces there are. The scout continues while there are unexplored interfaces in its map. The scout contracts with each site to be informed of any new interfaces that should arise or expire.

The scout's data base is available to affiliated guide programs that advise how to travel between sites. It is natural for guides to deal in the reputations of the carriers that they select for their customers. Sites that claim to move data but do so unreliably are not long recommended by guides.

What's in a Name?

We wish to impose few standards here. It seems, however, that some standard internal way for programs to name sites would be widely convenient if not necessary. We propose that a public crypto key is suitable for fundamental site names. We propose that one simply generate a public-private key pair and use the public key as a name. Only the real owner is able to process messages meant for him. Others, seeing X's public key, could claim to be X, but could not reply to any messages meant for X. The first 32, 48 or 64 bits of an RSA public key form suitably unique and short nick-names for many purposes.

Name Servers

Naming seems to be a natural monopoly but it need not be. Hierarchical naming schemes ala the Internet serve many purposes. Names that a person could carry with her would also be useful. Naming services are closely allied with guide services.

A given name server can sell alphabetic names. There might be a US name server that manages to convince most people in the US that his is the only important name server and registering "L. Jones" there is to effectively own that name. One service of this server would be to map alphabetic names to public key names. It is interesting to imagine competing between name sellers. We have thought out this to a few levels of gaming and it seems stable. A possible plan is to sell name-key pairs at \$1 each. More likely is to charge $\$10^{8-n}$ where n is the length of the name. This is only a crude estimate of the market price.

Other Services

Carriers, guides and reputation systems are all information services. They all work for cash. Data base operators, computers and all sorts of other traditional information services fit in this same mold. We imagine here a few more specific services that might arise.

I want to integrate a function or factor a polynomial. I send the problem to a Mathematica server together with \$.05 and get the response in a few seconds. I probably get the answer but the response may be that \$.05 of computing yielded no answer but more might do the trick. For a while it will preserve the state of the computation for resumption if I wish to send more money. The response might alternatively be that it knows the answer but will only reveal it for \$.25.

There may be an interface to a real bank thru which you can deposit and withdraw small sums of cash from your account; a CyberATM, or better, CyberTeller.

I have a document in TeX format but I can only print Post Script files. There is a service that does the conversion for \$.005 per page.

Anonymous Forwarders: Some sites will advertise a public crypto. They will expect all or some of their input packets to be ciphered. If such ciphered packets request some delay before forwarding, the site serves as a *mix*. (David Chaum introduced the term mix to describe a network service that forwards mail after removing standard indications of the source, thus providing anonymous mail.) The amount of money allows statistical tracing but this may be alleviated by establishing a small account at the mix. (Anonymous mail is a complex social issue and the operator of such a mix may be subject to social pressures.)

Specialized computing facilities could sell their service this way. If I have a computation that takes 10^9 bytes of real ram for $2 \cdot 10^{10}$ operations, some Cray machine could run my code in 20 seconds for \$3. If the Cray were kept busy with such business it would provide a 100% profit for the operator. Note that I need not be deemed a researcher by the government.

Even if 10^9 bytes of storage are not required I may prefer to spend \$50 for 30 seconds of a big Cray to do ray tracing than 3 hours of delay on my own CPU.

Guides may offer datagram forwarding service to avoid computing an optimal path from source to destination. Perhaps specialized forwarding services would make more sense in this case but guides already have full time circuits to each site.

Incentives and Reputations

> It is not from the benevolence of the butcher, the brewer, or the baker, that we expect our diner, but from their regard to their own self-interest. We address ourselves, not to their humanity but to their self-love, and never talk to them of our necessities but of their own advantages.
>
> Adam Smith, *An Inquiry into the Nature and Causes of the Wealth of Nations,* 1776

A site in this world can default, but when one buys an information service it is almost always caveat emptor. We do not request our money back when a magazine proves uninteresting. There is seldom a way for a third party to judge whether the service satisfied some contract. We rely thus on the positive reputation of the vendor.

Communication links fail which will cause interfaces to go down. A site may use a circuit as a detour for the interface. It might build the circuit in advance as a precaution. It might take a loss to maintain its reputation. Alternatively it is remarkably inexpensive to build two circuits between the same two endpoints, the second to serve as a hot standby. Since no data moves over the standby the cost is probably negligible. Circuits with a standby could arrange for their site agents to return the buffered data at the several sites to be retransmitted over the standby circuit.

When a neighboring site itself goes down it is harder to play fair. Only the failed site can properly interpret packets that were meant for it. It behooves a site X to arrange for at least one of its neighbors to notify routing services when X dies. This minimizes damage to X's reputation. It is in the neighbors interest to do so because, to an extent, their reputations rise and fall together. The whole neighborhood's reputation can suffer.

The concept of liability helps think about how systems like these work. When a site accepts a packet requesting forwarding to another interface it has acquired a liability or obligation. If it loses the packet it has defaulted but gains the amount of money in the lost packet. The site guards the packets to guard its reputation. We specifically reject mechanisms that would create an externally registered liability for each packet as that would likely cost more than the transaction itself. In particular there would be no legal liability. In the presence of real time automatic reputation systems it is enough that reputations can be lost in minutes.

Reputation Systems

Reputation systems can be built to provide current information on the performance of a site as observed by its customers. Examples of reputation services are Dun&Bradstreet, the local Better Business Bureau and Consumer's Union. Reputation systems are complex. We do not anticipate automating them completely. They must gain the trust of their customers and the only way to do this is to provide useful guidance to those customers. Indeed a reputation system must protect its own reputation.

Examples of such services are Dun&Bradstreet, the local Better Business Bureau and Consumer's Union.

Reputation systems are in a natural position to provide Yellow Page service.

Reputation systems can evaluate and rate a service by anonymously subscribing to it. Usually, however, they must evaluate complaints that come from customers of the rated services because the service is too expensive or it requires subjective judgment, or judgment by experts, who are, after all, the customers. The reputation service must thus judge the veracity and incentives of its reporters, namely the very customers of the reputation service. This requires the difficult skills of a diplomat, but ultimately incentives pull in the right directions.

There is an economy of scale in reputation systems. A user learns of the reliability of a service from other users thru reputation systems. He does business with many vendors according to their reputations as reported by the reputation system. He does business with only a few reputation systems and can thus directly judge these familiar systems.

Standards

This whole exercise is at attempt to avoid standards as if standards were bad. Actually standards are good but systems are sometimes bad because they cannot adapt to new standards. When we describe standards it is in part to establish that standards are possible in a world such as we suggest. Our main effort is to show possibilities if money is included in low level protocols. Behind each protocol description there is the suppressed comment that it could work some other way and work at the same time in the same network albeit with resulting inconvenience.

Bridges

To connect with classic networks we propose bridging sites that conforms to both worlds. The bridge keeps accounts for whoever needs to access CashNet from the Internet. If someone in the Internet needs to build a circuit to Y in CashNet then she must have an account with a bridge operator. She may then build a circuit to the bridge, identify herself somehow, and build a cash net circuit from there to Y. Her bridge account supplies money for packets going into CashNet and accumulates money for packets of hers coming back out.

There might be a collect call service where the bridge on its own speculative expense calls someone in CashNet in response to a query by an Internet citizen unknown to the bridge. The bridge asks if the CashNet citizen is willing to accept a collect call.

Calls from CashNet are easier. No account is needed for connection service to the Internet which can be paid for immediately.

Advantages

Decentralization affords diversity. Diversity fosters evolution. The protocol is simple and makes it possible to start up small businesses without the normal high cost of locating customers and arranging contracts.

Many one person companies. Many people already make a living indirectly thru the Internet. To do so requires finding some sponsor that will hire you with an motivation to provide some information service for free. The sponsor may indirectly be some government with a charter to provide service, or it may be an enterprise that supports collaborative research. Most information commerce cannot conform to these patterns.

Wide variety of services. Most current Internet services are currently free to the end user. This may be changing already but the Internet provides no integrated way of charging for such service. It is thus infeasible to sell some service whose intrinsic worth is only a few cents. When I wander into a book store and choose a book and pay for it with cash I need not learn the name of the owner of the book store nor need he learn mine. It is even easier when I buy something from a vending machine.

No Junk Mail! One could protect himself in a satisfying way if one advertised his reading tariff along with his net-name and directed his mail deamon to discard mail with insufficient tariff, then at least one would be compensated for reading such mail. More politely one's mail deamon could return the excess postage with a note that the message was automatically discarded since the worth was insufficient to justify reading the mail. One would surely get off of most junk mail lists that way! People with 900 series phone numbers get few solicitations! We don't mean this suggestion in all seriousness but we do hint at some social issues that are not exclusively electronic. Some "junk mail" may be desired by the recipient. We may want to see colloquium announcements or movie reviews. I instruct my deamon to watch for such announcements and return postage to the distributor sufficient to deliver the next package.

Availability of trivial services. I once got a total stranger to convert a TeX file to a PostScript file for me. The Internet is friendly place but I cannot continue to impose on strangers. If the converter were paid a penny a page he might find it worth his while to automate this service.

Communication systems have long been thought to be natural monopolies. To the technical obstacles to progress are thereby added bureaucratic and political obstacles. Technical progress has eliminated most of the original reasons for viewing the phone business as a natural monopoly, especially as cable companies install high bandwidth service to the home. When government operates phone systems as in many countries today, it has the political mandate, or feels it has, to establish policies that limit the uses to which lines may be put. Some of these policies are in fact to protect the established system from competition. Some countries were a decade or so late in digital communications because such service threatened conventional revenues. This is characteristic of monopolistic behavior. Government regulation is supposed to limit monopolistic manifestations. This is very difficult, perhaps impossible, even for intelligent uncorrupt non-political regulators.

Perceived Problems

Potential problems arise when we make fundamental changes to any working system. We examine several here and tend to argue that they will be transient.

Without central administration there is more opportunity for substandard service. Indeed there are no centrally mandated standards. This system would naturally encourage voluntary and de facto standards. Reputation systems would report compliance with such standards. Mandated standards are often unachieved. Competition even solves the problem of what the standards should be—How good is good enough?

Without central administration there are greater opportunities for fraud. It will bother some that forwarding sites have neither accountability nor legal liability. Operators of such sites must plan to make their living from repeat business. Just as a restaurant must provide food acceptable to its customers to ever make a profit, so must such sites play largely by the "rules". One type of fraud is especially egregious—"losing" packets and pocketing the money. This is especially tempting since one may be able to blame the loss on your neighbor. Indeed this aspect bars application of CyberCash from transfers of large sums. Just as one normally does not carry $10,000 as he wanders through strange neighborhoods, neither does one send $10,000 packets thru most sites. (This is another justification of the 32 bit money field denominated in $.00001) Repeated packets, with acknowledgments, could move large sums but this may run up against limits imposed at interface accumulators where the credit of one site operator may be insufficient in the eyes of the downstream operator. It all boils down to the fact that big money is a big deal, even for banks! Incidentally this problem with big money is better handled with schemes such as Chaum's digital money which involves a bank and does provide accountability and perhaps legal liability.

Protection against gridlock may require some distributed logic. Gridlock occurs when some set of sites are unable to accept new packets for lack of storage and all of the packets that they need to deliver (to relieve their congestion) are directed to members of the set, none of whom are receiving. We have not thought out an incentive based solution to this yet. Circuits alleviate but do not eliminate this problem. This difficult exercise is left for the reader

Operator Credit

These concerns involve both the individual end user whose personal computer implements this protocol or some huge communications carrier.

If the credit of one of the site operators at an interface is insufficient for the other operator then the latter may establish a limit on the accumulator beyond which traffic is not accepted. This would likely be the case for end-users that would thus be limited in the bills they could run up. The credit limit may be zero, thus requiring prepayment. Conversely the same limit protects the user from running up unlimited bills thru misunderstanding of the software that handles his cybercash.

Perhaps such interfaces would make EFT (Electronic Funds Transfer) arrangements with normal financial institutions when some economic amount of net money is involved or credit becomes stretched. This is an ideal application for Chaum's DigiCash.

Other Issues

Several ideas have been omitted from this paper that concern how a network of this sort can provide a degree of security and reliability beyond what is possible to achieve based on the Internet protocols. These issues are somewhat orthogonal but may, nonetheless, be strategic to each other. We expect to produce another paper covering these ideas soon. We hope to speculate in another paper about prices that might be achieved if there is free entry by communications providers into this market.

Conclusion

Much of the charm of the Internet is the informality of access. One can learn of a program thru netnews, fetch it via FTP and run the program in the span of an hour with no bureaucrats or clerks involved and certainly no lawyers. This is as much a matter of Internet culture as charter. But what has more charm and informality than a flea-market where cash and anonymity prevail?

Bibliography

Braun, Hans-Werner, and Claffy, Kimberly C., and Polyzos, George C.,
"A Framework for Flow-based Accounting,"
to appear in Proc. Singapore International Conference on Networks (SICON'93), Singapore, September 1993.
(Also available by anonymous ftp from
ftp.sdsc.edu:pub/sdsc/anr/papers/accting.sg.ps.Z)

Brands, Stefan, *An Efficient Off-Line Electronic Cash System Based On The Representation Problem.* Anonymous ftp at:
ftp.cwi.nl:/pub/CWIreports/AA/CS-R9323.ps.Z

Chaum, David. Achieving Electronic Privacy, (blind signature technology) *Scientific American* v267, n2 (August, 1992)

Dukach, Semyon, SNPP: A Simple Network Payment Protocol. *Proceedings of the Eighth Annual Computer Security Applications Conference*, december 92. Anonymous ftp at:
ana.lcs.mit.edu:pub/snpp.

Ferguson, Niels, *Single Term Off-Line Coins*, in "Proceedings of EuroCrypt'93", 1994, To appear. (Anonymous ftp at:
ftp.cwi.nl:/pub/CWIreports/AA/CS-R9318.ps.Z)

Huberman, B.A., editor, *The Ecology of Computation* (North-Holland, New York, 1988).

MacKie-Mason, Jeffrey K., and Varian, Hal R.(1993). *Some Economics of the Internet.*
gopher: gopher.econ.lsa.umich.edu

MacKie-Mason, Jeffrey K., and Varian, Hal R.(1993).*Pricing the Internet.*
gopher: gopher.econ.lsa.umich.edu

Pool , Ithiel de Sola, *Technologies of Freedom*, (Belknap Harvard, Cambridge MA).

Temin, Peter with Louis Galambos. *The Fall of the Bell System, a study in Prices and Politics.* Cambridge, Cambridge University Press, 1987

tpub.parts.d3 96.11.13

KEIO TRANSPUBLISHING PROJECT

We are working on a number of aspects of electronic publishing necessary for tomorrow's on-line document systems.

1. TRANSCOPYRIGHT: a form of permission allowing people to re-use and republish material virtually on the network."Trans©" is a compact way of saying, "I permit you to re-use this material in some new context, if it is delivered from my server." This allows credit and payment to the originator.

See our Web pages at

`www.sfc.keio.ac.jp/~ted/`

2. But transcopyright leaves a lot of options for the rightsholder. The proposed "**KEIO TRANSCOPYRIGHT FORMATS**" will provide suggestions for various specific details of transcopyright permission.

This will include a simplified permission statement and a transcopyright notice divided into two pieces, one pointing to the permission statement and one pointing to the original context. It will also provide suggested transcopyright formats for other data types, especially sound and video.

3. A TEXT SOURCE TAG. Already transcopyright can be directly used on the World Wide Web for GIF and JPEG pictures, using the standard tag

`<IMG SRC = ...>`

We are now implementing an HTML tag allowing TEXT to be quoted-- not just pictures-- using a format corresponding to the above:

<TXT SRC = . . .>

While our implementation will not automatically add this tag to the standard html system, it is a first necessary step toward such adoption.

4. MICROPAYMENT ON TOP: We are presently working with ASCII Corporation on a system for micropayment. Ideally such a system could be added to any Web service without changing that service. We are investigating to see if such a payment system is possible.

5. OSMIC STRUCTURE. This is a new proposed data structure, Open Standard for Media InterConnection, which will allow the publication of complex, stable media objects whose links do not break as the object changes.

6. PUBLISH-BY-NAME (URN). Today, publication on the Web makes a document synonymous with its address (URL). This is unfortunate. A "publish-by-name" system will give the object a unique nameplex (title + author + time + node of origin), sometimes called a Universal Resource Name or URN. This will allow the object to be put at many different locations, both for efficiency of delivery and safety of the contents.

7. STABLE PUBLISHING WITH LONG-TERM COMMITMENT. Today's Internet publishing is temporary and unreliable; there is no way to make a believable promise that a document will be available for 50 years or longer. We are investigating the organizational arrangements that will make

TRANSPUBLISHING

CAN IT HELP SOLVE THE COPYRIGHT PROBLEM?

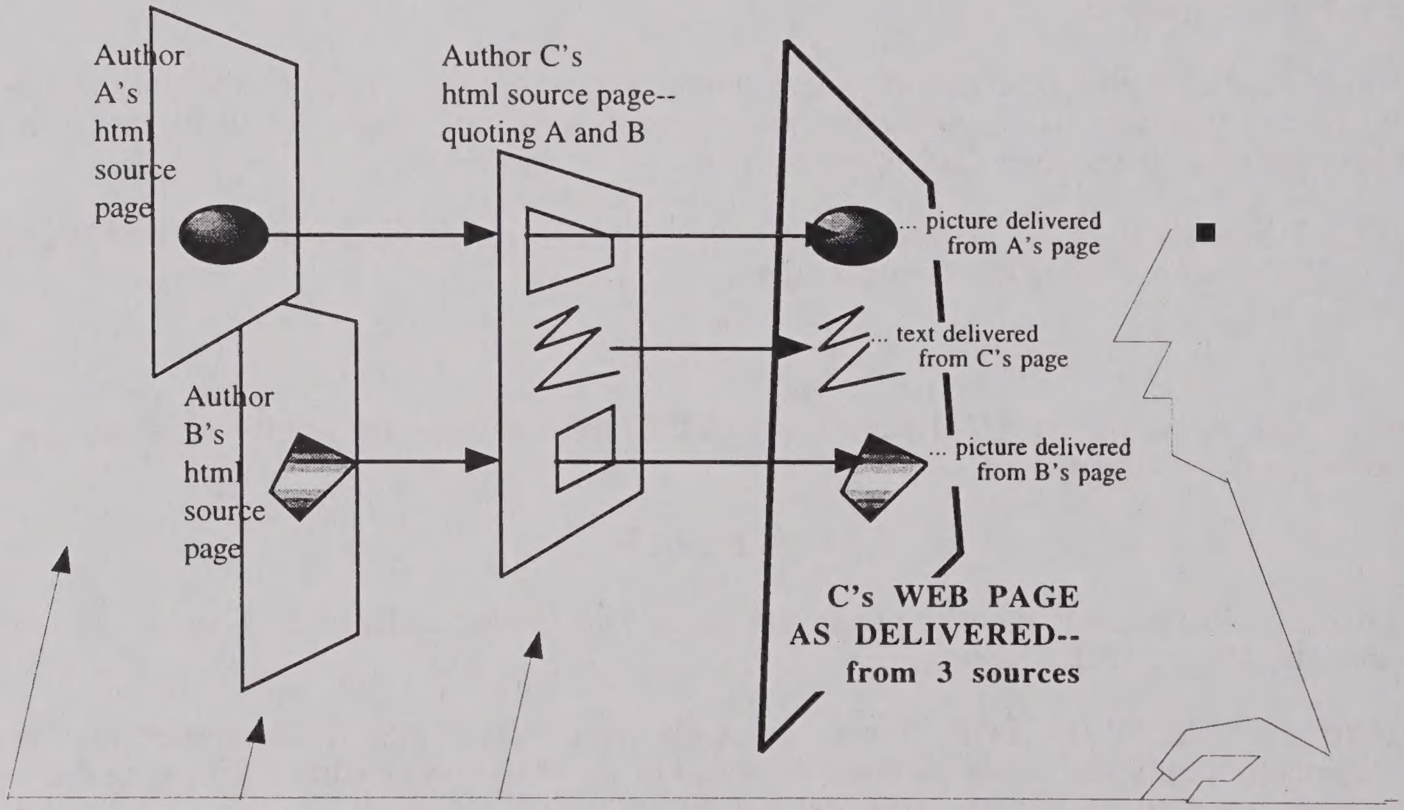
The copyright system creates a conflict between two desirable ideas.

(1) Authors, artists and publishers want praise and profit for their writings, pictures and other creative work.

But (2) the public wants to be able to re-use copyrighted materials in ways that they find beneficial and appropriate.

TRANSPUBLISHING is a possible Win-Win solution that combines both desirable features: contents may be freely re-used and republished in new arrangements by anyone, while the authors and publishers receive credit (and possibly money). It only works on computer networks (like the Internet).

In transpublishing, an electronic document quotes from another electronic document by pointing at the part to be quoted. Each quotation is delivered to the user from its original (or some equivalent, owned by its originator). This means that the user's payment for that part (if there is any payment) goes to the originator.


Author A's html source page
Author C's html source page-- quoting A and B
... picture delivered from A's page
... text delivered from C's page
Author B's html source page
... picture delivered from B's page
C's WEB PAGE AS DELIVERED-- from 3 sources

By user's choice of C's page, user automatically pays A, B and C a very small amount for their pieces.

TRANSCOPYRIGHT is a new legal arrangement, a form of permission allowing people to re-use material in this way on the network. "Trans©" is a compact way of saying, "I permit you to re-use this material in some new context, if it is always delivered from where I say." This allows the material to be re-used in new contexts with credit and payment to the originator.

(1)

Confidential.
mint.n.xu.briefing.d5
96.11.30

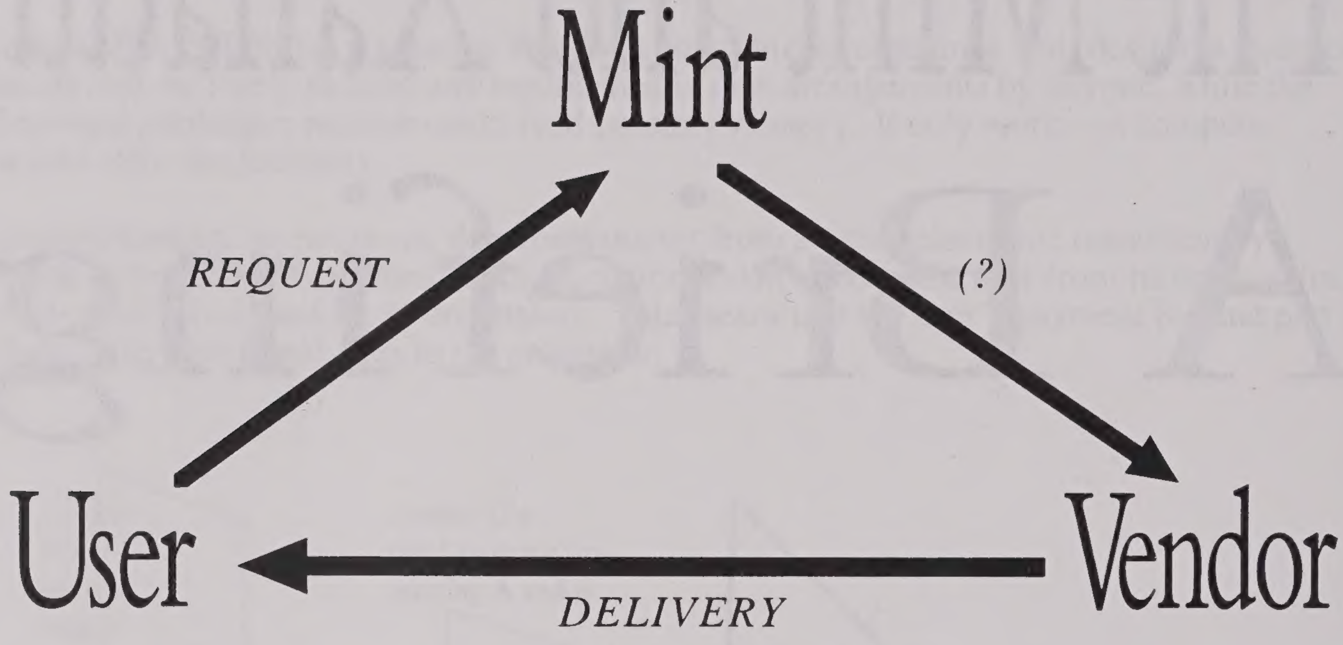
The Bounce Method, The Mint and Xanadu®: A Briefing

Theodor Holm Nelson

Third HyperCoin Workshop
96.11.30-96.12.01

mint.n.xu.base.d3
96.11.29

BOUNCE METHOD IN GENERAL



1) The user's Web request goes first to the Mint.

2) The Mint does something or other.

3) The Mint then bounces the session to the Vendor.

4) The Vendor delivers the goods (sometimes a Web page, sometimes not).

THE DIFFERENT BOUNCE METHODS ARE VARIANTS OF THIS BASIC PATTERN.

We will look at these variations in succession.

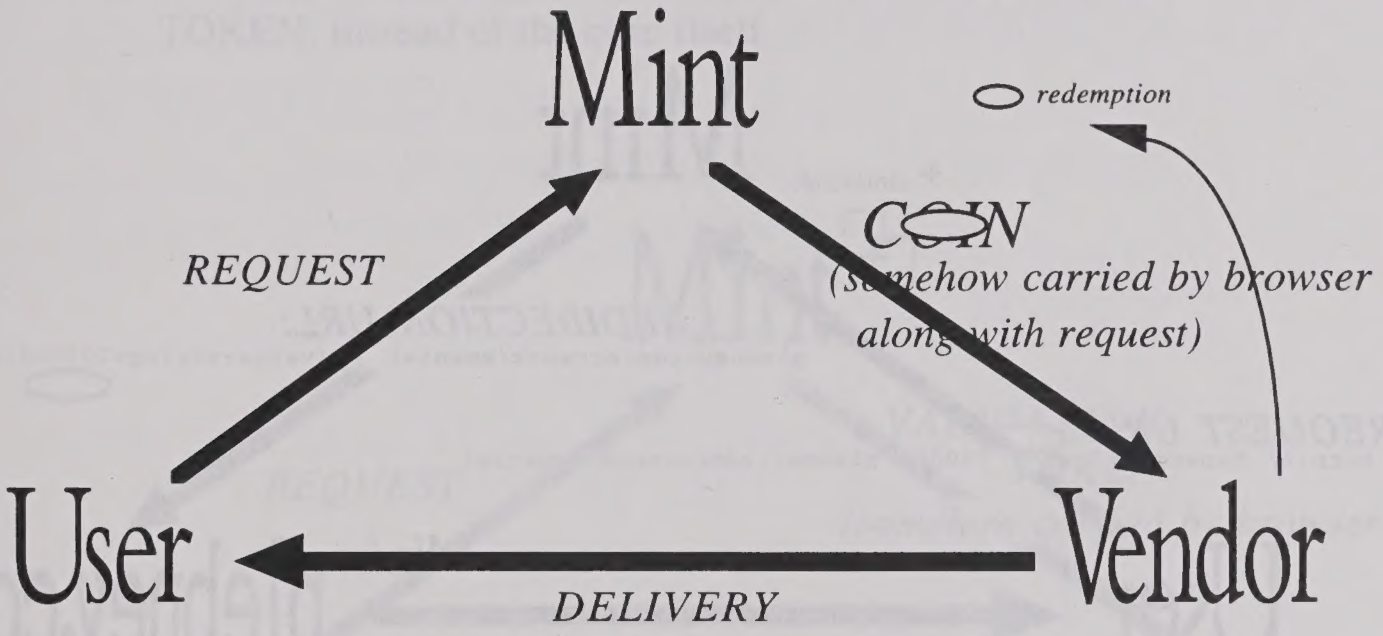
Confidential.
mnx.boz.simp.d4
96.11.29

BOUNCE METHOD WITH COINS

(distributed throughout the Net)

The user's request goes to the Mint. The Mint creates a "coin," some sort of encrypted value string. (Specifics below.)

Coins move all around the Net; thus it is a "distributed-coin" system.


Mint
redemption
REQUEST
COIN
(somehow carried by browser along with request)
User
Vendor
DELIVERY

1) The user's Web request goes first to the Mint.

2) The Mint makes a "coin" of the appropriate denomination, which it gives to the User's browser.

3) The Mint then bounces the session to the Vendor.

4) The Vendor tests the coin for apparent validity, then delivers the goods.

5) Vendors return the coins to the Mint for redemption and credit of their accounts.

④

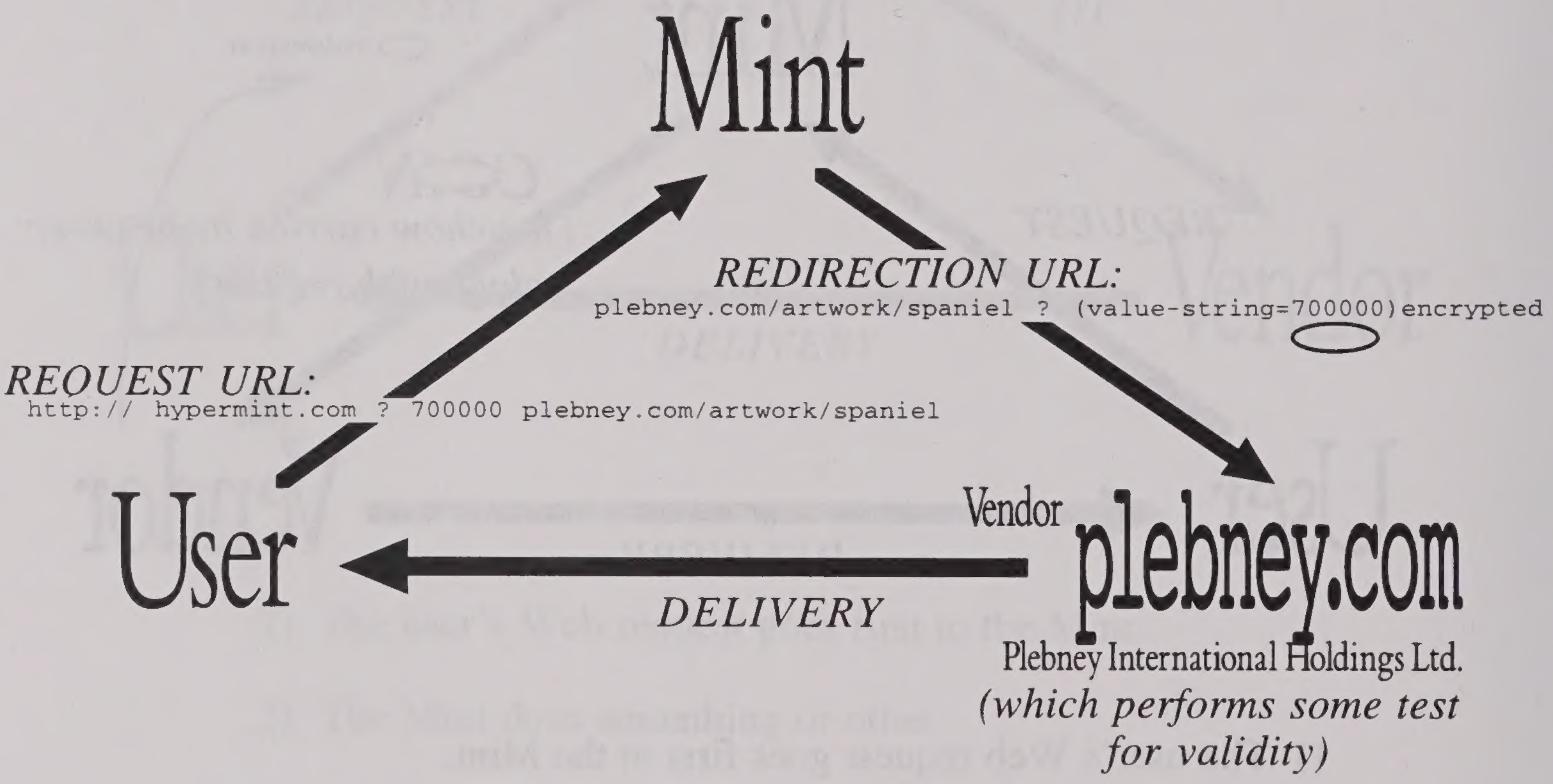
Confidential.
mnx.boz.simp.detl.d5
96.11.29

BOUNCE METHOD WITH COINS [detail]

The publisher creates a URL that goes first to the Mint, names the price, and THEN states the real URL of the material.

The Mint replaces the URL with the real URL of the material, then adds the value string-- in the URL itself.

This is called a "checklet."™


Mint
REDIRECTION URL:
plebney.com/artwork/spaniel ? (value-string=700000)encrypted
REQUEST URL:
http:// hypermint.com ? 700000 plebney.com/artwork/spaniel
User
Vendor
plebney.com
DELIVERY
Plebney International Holdings Ltd.
(which performs some test for validity)

The main presumption is that

**THERE IS MUCH LESS OVERHEAD
if
ALL THE VENDOR NEEDS TO TEST
is
WHETHER THE COIN CAME FROM THE MINT.**

The Mint is guaranteeing the coin; the Mint can worry about the User.*

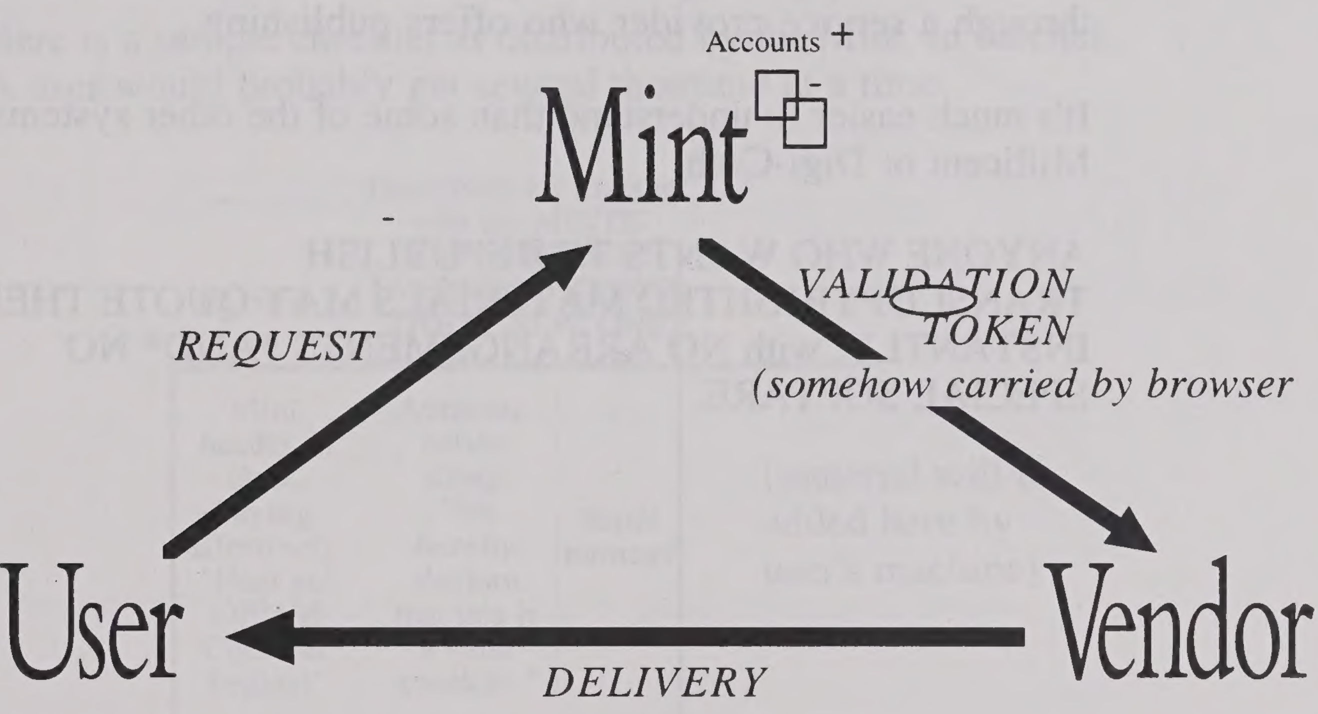
(Cf. "I don't have to outrun the bear, I just have to outrun *you*..")

BOUNCE METHOD WITH ACCOUNTING

The publisher creates a URL that goes first to the Mint, names the price, and THEN states the real URL of the material.

The Mint replaces the URL with the real URL of the material, then DECREMENTS THE USER'S ACCOUNT, INCREMENTS THE VENDOR'S ACCOUNT.

The Mint then redirects the session with a VALIDATION TOKEN, instead of the coin itself.


Accounts +
-
Mint
REQUEST
VALIDATION
TOKEN
(somehow carried by browser
User
Vendor
DELIVERY

This has some advantages. It does involve the Mint more directly in transactions, which could be a legal detriment.

(6)

Confidential.
pow.advantages.d1
96.11.30

POWERFUL ADVANTAGES
of the Bounce Method.

The system uses Ordinary Web Browsers.

Users/purchasers may participate INSTANTLY, with NO SOFTWARE INSTALLATIONS; just an account.

Publishers may participate easily, either with a CGI script or through a service provider who offers publishing.

It's much easier to understand than some of the other systems, like Millicent or Digi-Cash.

ANYONE WHO WANTS TO REPUBLISH TRANSCOPYRIGHTED MATERIALS MAY QUOTE THEM INSTANTLY, with NO ARRANGEMENTS *AND* NO SPECIAL SOFTARE.

Confidential.
best.coin.for.bouncing.d3
96.11.30

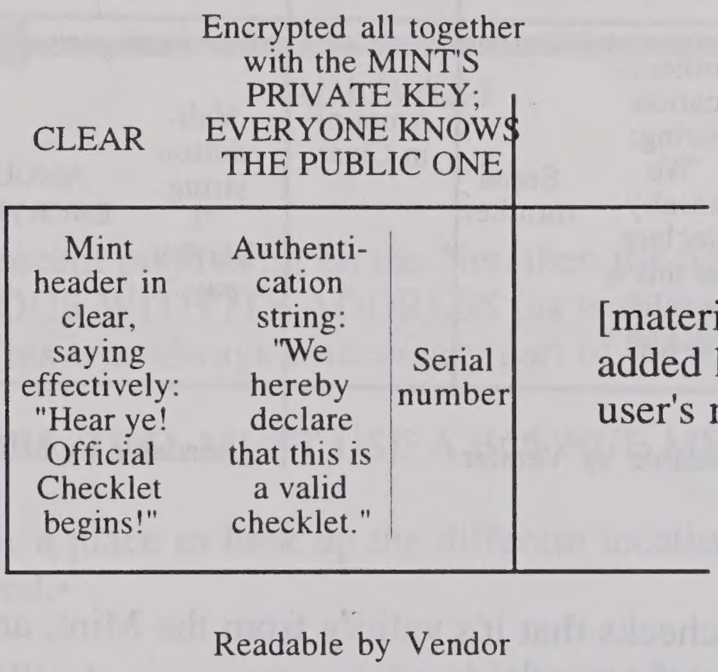
COIN MODEL: Best So Far

A number of distributable coin designs have been proposed. Here is the best one so far.

The Mint distributes* blank checklets to its paying customers, and a program to use them automatically if the prices are within acceptable range.

THE VENDOR WILL BE ABLE TO VERIFY THAT THE CHECK IS FROM THE MINT; THE MINT WILL BE ABLE TO VERIFY THAT THE CHECKLET IS AUTHENTICALLY FROM THE CUSTOMER IT WAS DISTRIBUTED TO, AND REPRESENTS A VOLUNTARY PAYMENT (as defined within the rules of this system).

Here is a sample checklet as distributed by the Mint, in batches. A user would probably get several thousand at a time.


Encrypted all together with the MINT'S PRIVATE KEY; EVERYONE KNOWS THE PUBLIC ONE
CLEAR
Mint header in clear, saying effectively: "Hear ye! Official Checklet begins!"
Authentication string: "We hereby declare that this is a valid checklet."
Serial number
[material will be added here by user's machine]
Readable by Vendor

---

* Sells or gives; this is irrelevant to the method.

(BEST COIN, COINTINUED.)

The Mint has a public key everyone knows; only the Mint knows its private key.

Each user deposits with the Mint a different key.*

The user's machine fills in the amount of the coin, encrypts it with his private key, and can send it directly to the Vendor (bypassing the Mint, which has authenticated it because it trusts the user).

CLEAR	Encrypted all together with the MINT'S PRIVATE KEY; EVERYONE KNOWS THE PUBLIC ONE		CLEAR	Encrypted with USER'S PRIVATE KEY, KNOWN TO MINT	
Mint header in clear, saying effectively: "Hear ye! Official Checklet begins!"	Authentication string: "We hereby declare that this is a valid checklet."	Serial number	Amount in Clear	Validation string: "I hereby pay."	AMOUNT ENCRYPTED
Readable by Vendor				Readable by MINT ONLY	

The Vendor checks that it's validly from the Mint, and can therefore send the goods.

In due course the Mint receives the checklet and can review that it was validly signed by the user.

Q.E.D.

---

* This need not be a trapdoor code, but any fast key.

Confidential.
urn.lookup.d5
96.11.29

The Compelling Resemblance of the Bounce Payment Method to--

"WHERE IS A COPY OF THIS DOCUMENT?"

-- essential for decent network publishing.

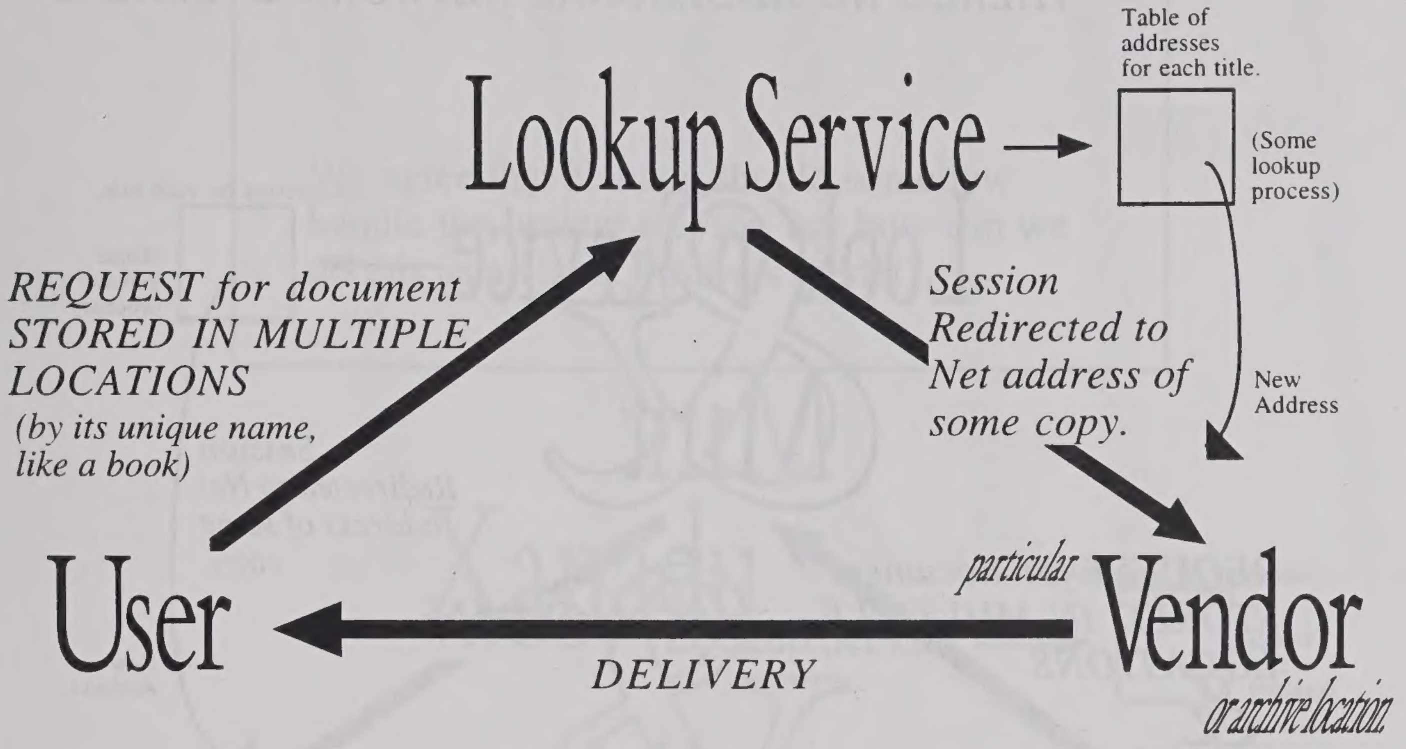


Table of addresses for each title.
Lookup Service
(Some lookup process)
REQUEST for document STORED IN MULTIPLE LOCATIONS
(by its unique name, like a book)
Session Redirected to Net address of some copy.
New Address
User
particular
Vendor
or archive location
DELIVERY

If there is to be decent publishing on the Net, then the document MUST NOT BE SYNONYMOUS WITH ITS ADDRESS (as with the current ridiculous URL system). This was always a necessary part of the Xanadu design.

NOTE THAT THIS, TOO, MUST USE A BOUNCE METHOD.

There needs to be a place to look up the different locations where the document is stored.*

The original WWW design proposed that this would be called "URN" (Universal Resource Name) or "URL" (Universal Resource Locator) or "URI" (Universal Resource Identifier).

I prefer to call it "Publish-by-Name," which the public will probably like best. Everybody is weary of the url mess, which we may call Publish-by-Address.

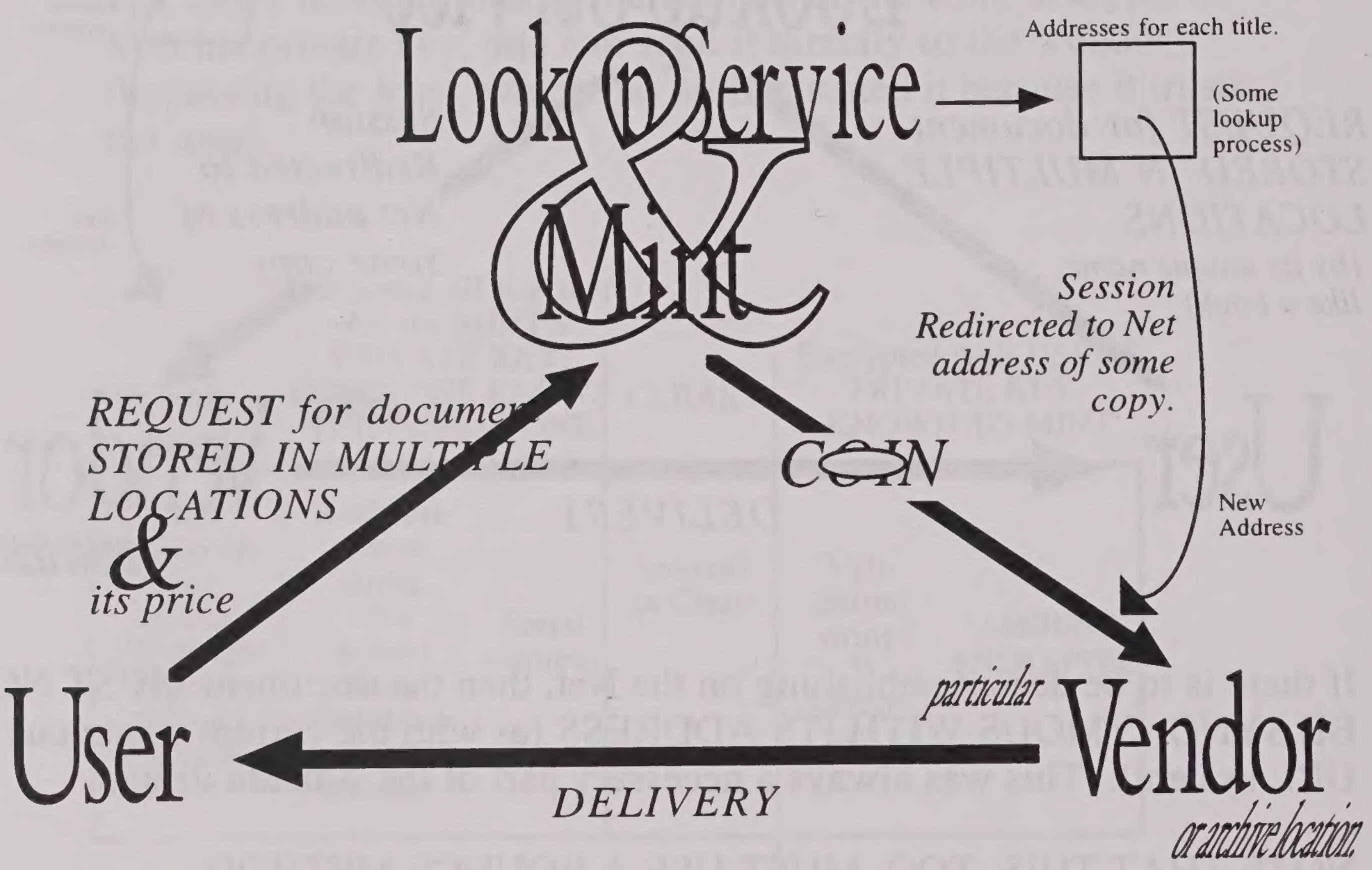
---

* (It bears some similarity to the DNS lookup of Internet domains, except that DNS is 1-to-1 and this must be 1-to-many.)

The obvious thought--

WHY NOT COMBINE THEM?

THERE'S NO ADDITIONAL NETWORK OVERHEAD!


Look up Service
&
Mint
Addresses for each title.
(Some lookup process)
Session Redirected to Net address of some copy.
New Address
REQUEST for document STORED IN MULTIPLE LOCATIONS
&
its price
COIN
User
particular
Vendor
or archive location
DELIVERY

See here.

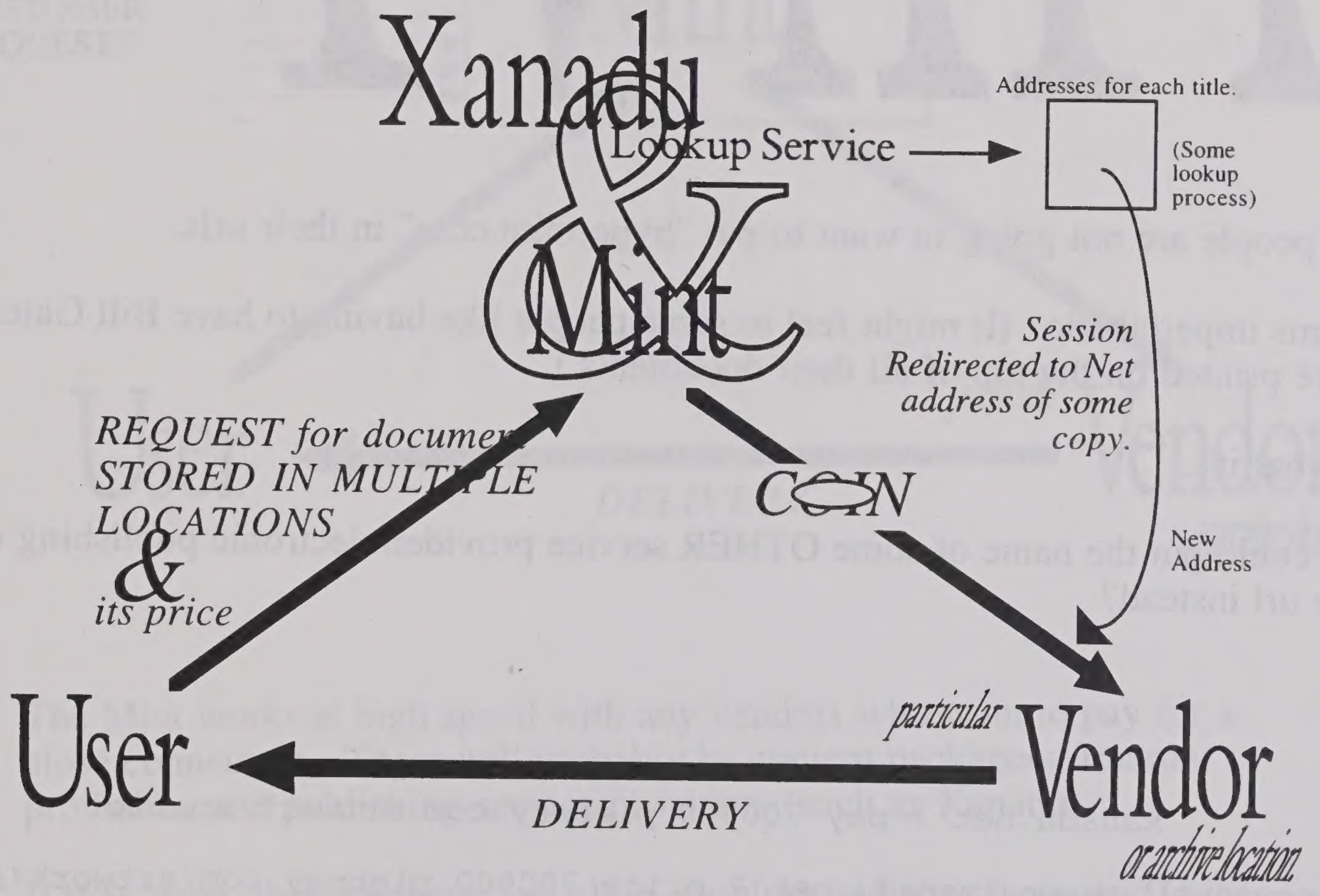
If we combine these two services, and do them in parallel, it'll take no longer than if we did only one of them.

SYNERGY!

But what should be the exact arrangement?

But HOW Combine Them?

We agree that Xanadu should somehow handle the lookup service, but how can we do this as a viable business model?


Xanadu
Lookup Service
&
Mint
Addresses for each title.
(Some lookup process)
Session Redirected to Net address of some copy.
New Address
REQUEST for document STORED IN MULTIPLE LOCATIONS
&
its price
COIN
User
particular
Vendor
or archive location
DELIVERY

Confidential.
hint.dl
96.11.30

HINT

Most people are not going to want to put "hypermint.com" in their urls.

It seems imperialistic. (It might feel to some people like having to have Bill Gates' picture printed on the top of all their documents.)

But what if ...

They could put the name of some OTHER service provider/electronic publishing company in the url instead?

Like

`xanadu.net ? pay 700000 plebney.com/artwork/spaniel`

`glasswings.com.au ?xanadu.net ? price 700000 plebney.com/artwork/spaniel`

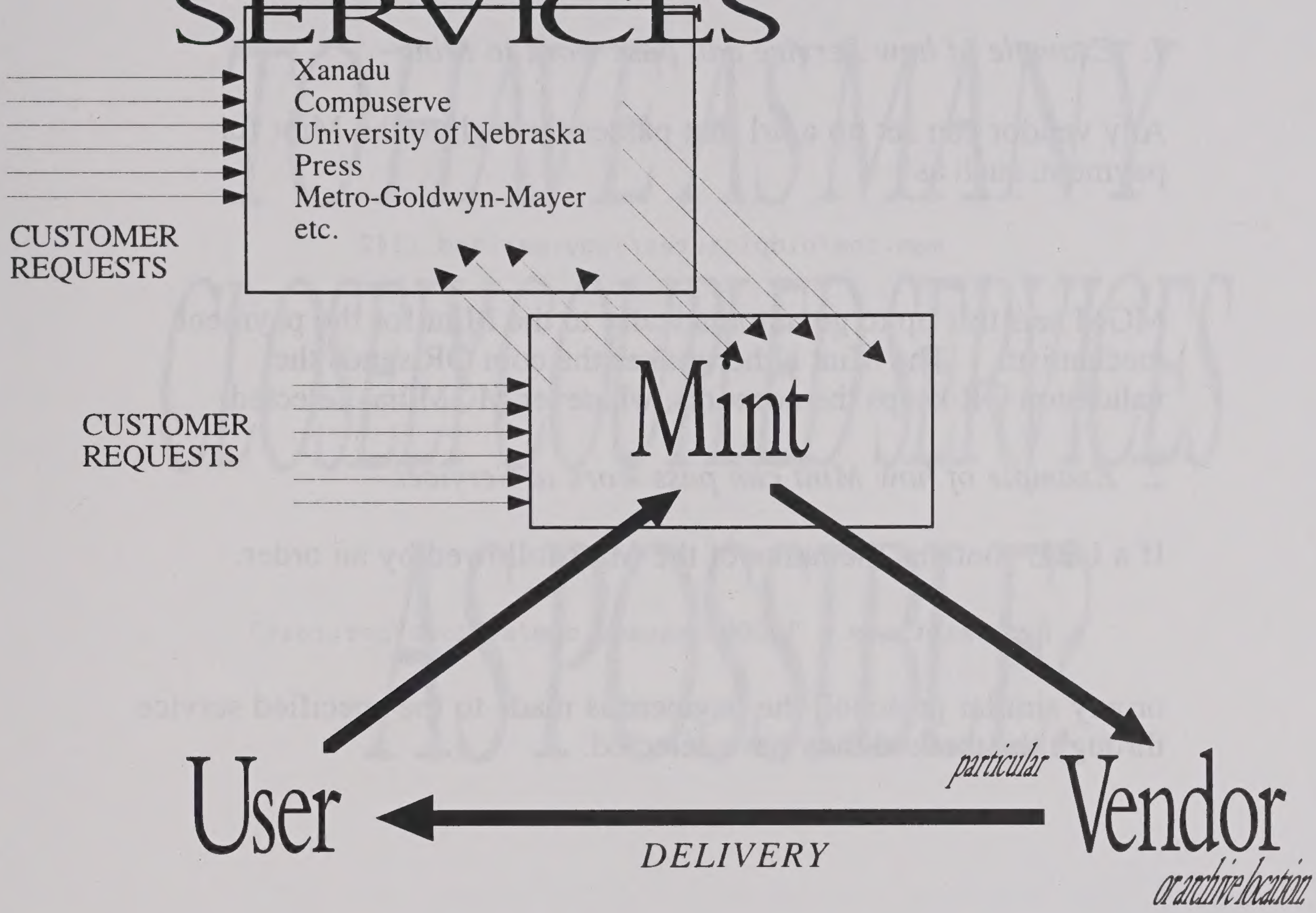
`xcompuserve.com ? hypermint 700000 plebney.com/artwork/spaniel`

Any of these could connect to the Mint ALSO to provide payment services through us, with CGI scripts that do the same thing we are planning to do with

`hypermint.com ? 700000 plebney.com/artwork/spaniel`

Confidential.
close.coupling.d2
96.11.30

CLOSELY COUPLED SERVICES


Xanadu
Compuserve
University of Nebraska Press
Metro-Goldwyn-Mayer
etc.
CUSTOMER REQUESTS
CUSTOMER REQUESTS
Mint
User
particular
Vendor
or archive location
DELIVERY

The Mint works at high speed with any vendors who want to pay for a close connection. These will probably be content packagers, Internet providers, and publishing-service providers (such as Xanadu).

If they want, they can have fast close coupling and pass requests TO the mint, get requests FROM the mint as specified:

Traffic arriving in either diretion is quickly passed to the other side.

14

Confidential.
coupling.examples.d4
96.11.30

CLOSE COUPLING:
Examples.

1. Example of how Service can pass work to Mint--

Any vendor can set up a url that passes through to the Mint for payment, such as

`mgm.com/oldpictures/judy.garland.1142`

MGM sets this up to go automatically to the Mint for the payment mechanism. The Mint either makes the coin OR sends the validation OR keeps the accounts, whatever MGM has selected.

2. Example of how Mint can pass work to Services--

If a URL contains the name of the Mint followed by an order,

`hypermint.com ? 700000 xanadu.com/allison/portrait7`

or any similar protocol, the payment is made to the specified service through the method they have selected.

15

Confidential.
why.impt.dl
96.11.30

WHY IS IT IMPORTANT TO HAVE AS MANY CLOSELY COUPLED SERVICES AS POSSIBLE?

Confidential.
critical.mass.d2
96.11.30

CRITICAL MASS
The Launch Problem.

Success depends on getting critical mass: thousands of vendors and major service providers who want to go through us.

If adoption is slow, we can easily fail, because the Net moves so fast.

If adoption is slow, political opposition from the W3 committee will appear.

VISIBLE, FAST SUCCESS IS IMPORTANT.

Therefore WE WANT AS MANY PEOPLE CONNECTING THROUGH OUR METHOD AS POSSIBLE, AS FAST AS POSSIBLE.

17

Confidential.
summary.d1
96.11.30

SUMMARY AND CONCLUSIONS

We should implement a number of methods, the better to work with everyone's different needs.

The Bounce Method is a FAMILY of methods have incredible advantages, and can be adapted and tuned in various ways.

The appropriate coupling between the Mint and Xanadu becomes clear when we consider --

1. Lookup should be combined with bounce;

2. Thus Xanadu should be closely coupled;

BUT

3. The Mint should develop, from the idea of close coupling to Xanadu, a generalized system of close coupling with other services;

4. The more generalized this is, the better or chance of rapidly acquiring market share.

5. Xanadu should be given a certain amount of lead time to get established, after which it should simply be one of the client services closely coupled to the HyperMint.

Saturday
Whiteboard
Printouts

AGENDA

10 - 12 Review & Discussion
12 Open Formal Meeting

Banking Industry Experience
- Japan
- International
- Cash card.

Technology Issues
- Show Shoppers
- Crypto Issues.

Is Mint a public Service or For Profit Company?

Privacy & Data Mining

* BANKING is a local business.

Jack's Report (David)

Early Entry vs { strategic & tactical execution

Economics

Governmental Issues

Competition/IP

Kay Nishi - Views

Viability of HyperCoin

Demo & Implementation
  Client Side
  Server Side

Business Case
Business Model
{ Efficiencies
Flatten Value Chain)

Busine Organization

Future Projections
- Timing
- market
- Active Participation

Technology – Low cost transactions
Essence of our approach –
 - very low cost transactions.
 - all transactions could be done electronically

Banks don't like small customers.
Net – datagram cost $\Theta(\$0.000001)$?
Bank – minimum transaction $\Theta(\$0.50)$
JV – smart cybersurfs Need to deal with machine & agent transactions.
DA – Pay to advertise to me! ←
Very low cost, limited security transactions are interesting
97% cash transactions < $10

Banks are local entities
∃ no global bank
$$ transactions are heavily regulated.
Gold is prohibited as $-medium in Korea.
☆ Gold as a medium may not be reasonable in terms of modern economic models.
Need to examine need for a nanotransaction
Banking behavior is cultural.
Agent to Agent Transactions.

Nano * (* = {market, transactions, ...})

What is a nanotransaction?: (DA – low cost transaction – ~~amount~~ amount not relevant)

$ 10e+9 LARGE – B-B ← Not Nano
10e+6 SMALL CORP – Lg mov?
4 co

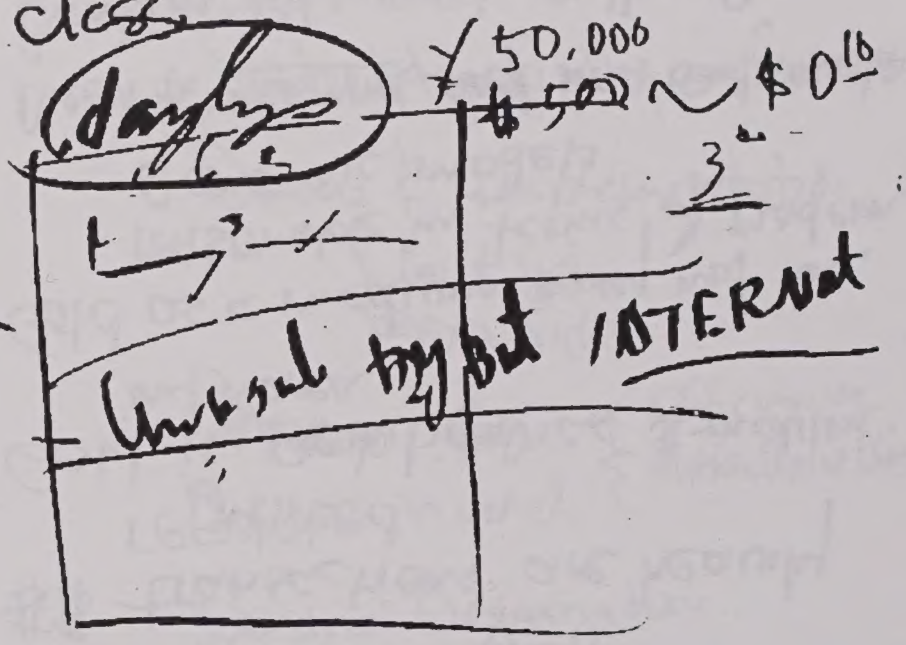
visa: 10 +3 appliance
10 +1 lunch

cash: 10 +0 newspaper . post card
10 -1
10 -2 xerox copy
10 -3 web biz

10 -6 IP datagram

Shiomura's Value Scale

Transaction Scale (value) & Trust define transaction class.



Allison/Gill Off-line

Constraints

> Advantage
> Benefits
> Efficiency (economy/market)
> Banks are natural market - off load small customers.
> Agents are real users, (smart agent time scale)
> Cost (accounting - infrastructure - aggregation - dispute resolution)
> Dynamic Pricing (Agoric Systems)
> Utility Co as model (Telephone, Water, Electricity, etc)
> " " " market

off-line
vs 'on'-line
pricing
issue

Nano * (* = {market, transactions, ... }

⊗ $\exists$ an aggregation cost which needs to be handled.

⊗ Granularity & minimum settlement amount are distinct concepts.

⊗ High settlement costs may define protocol to ensure an efficient market

⊗ BUSINESS SHOULD MAKE $ (e.g. Worldcash, $400,000 profit, $200K/year)
  - Simplification is important.

⊗ Need to match NEW economic landscape enabled by nanotransactions.

⊗ We may be a White Knight!

⊗ Internet is NOT FREE. (but appears to be)
  - Problem: how to move costs to users.
  - Current market is very inefficient.

Another important issue — non-physical nature of the product being produced. [Resend on failure] <Assumes product is a document/information. Why not use Nano-transactions for commodities?>

Recourse <What happens if bad info is relied upon? Liability & Recourse>

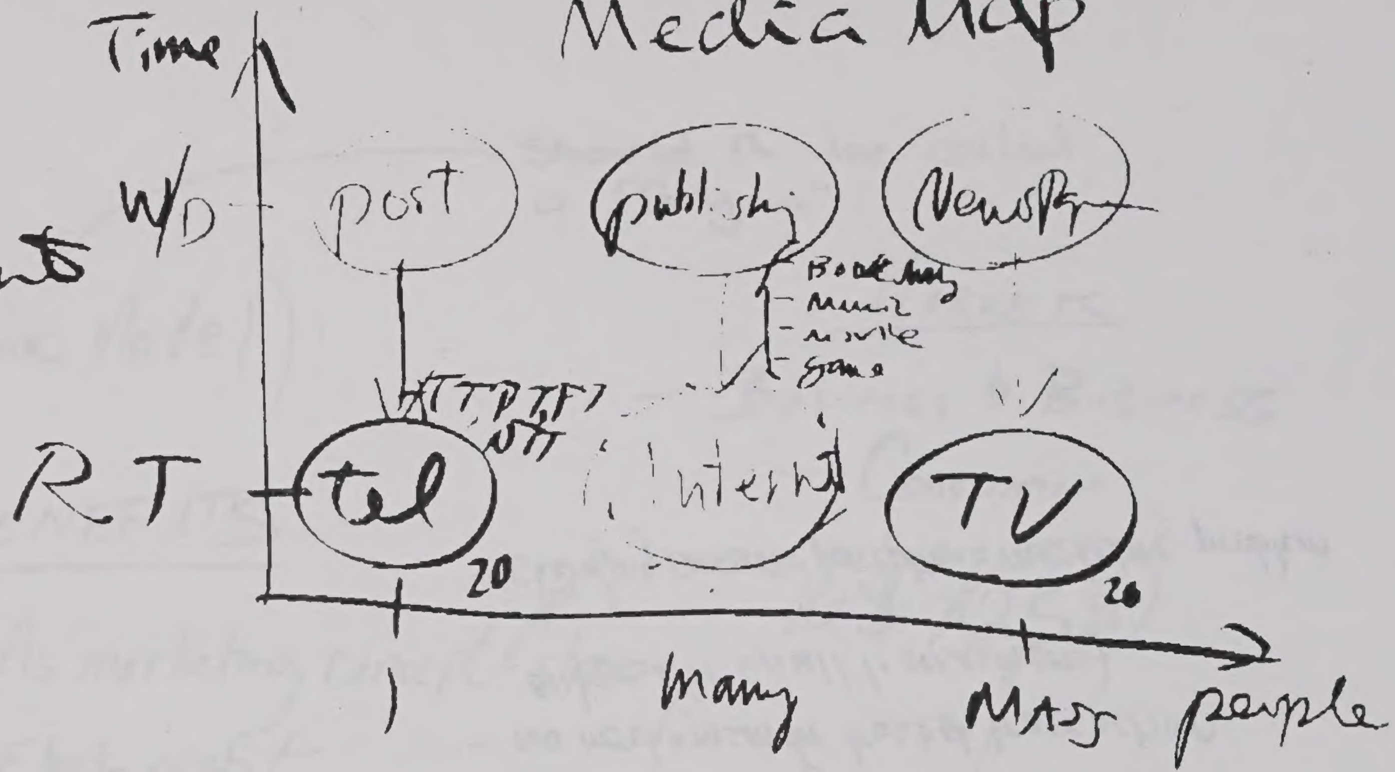
- Key Problem: Privacy and Data Mining (JV)
- Agoric Systems
- 128-bit IP Standard & cost negotiation
- Copyright Law and what is happening.

Kay Nishi

+ 20 years Ago
1995 - Internet

1. GOLD
2. JV or alone
3. Mint, Bank, Archives
4. don't touch Browser
-

Media Map


Time
W/D
RT
post
publishing
Newspaper
Book, Mag
Music
Movie
Game
NTT, DDI
NTT
tel
20
Internet
TV
20
1
many
MASS
people

Kevin Vaughn

The Saleable Product/Service?

(the micropayment prouduct/system)

1.

- PRIVACY
- No mechanism for trusted transaction online (for small transactions)
- Changing consumer perception = marketing problem

Should it be called a "Bryan"

1 GOLD (Economic Model)

MARKETS

- Business to Business
- Consumer

16$ / 25$

UES (Negatives)

1. Russia + S.Africa + Auss.
2. Denomination = Consumer comprehension / acceptance
3 Variable rate of gold
4 fulfills req's. for a "new uit of measure"
5. fully international
6. ?

7. Controversial

BENEFITS

1. As marketing concept - very strong
2. Float = profit
3. Gold has image of solidity
4. Market hedging / deriv's. = more earning opp's.
5. Controversial

1996 = $80.00 oz.

80's = 8__.00$

30$

Business Ownership (the Players)

Requirements for Success

Governments
Publishers
Libraries, Content Providers
Banks
Trade Org's.

Funding from:
ASCII

Mint Characteristics
1. Issues "Cards" (16/25ex)
2.

Bank Char's.
1. Distributor
2 Guarantor
3.

Options
1. Acquisition of exist. bank
2. Venuet) [Offshore]
3. JV's / Partnerships
4 Other key relationships
5. Mint

1. How to mitigate competitive risk of established financial institutions?

Wholly Owning a BANK

NEED LOCAL MAJOR
CORRESPONDING BANK

ISSUES

1. (U.S.) Regulatory Issues

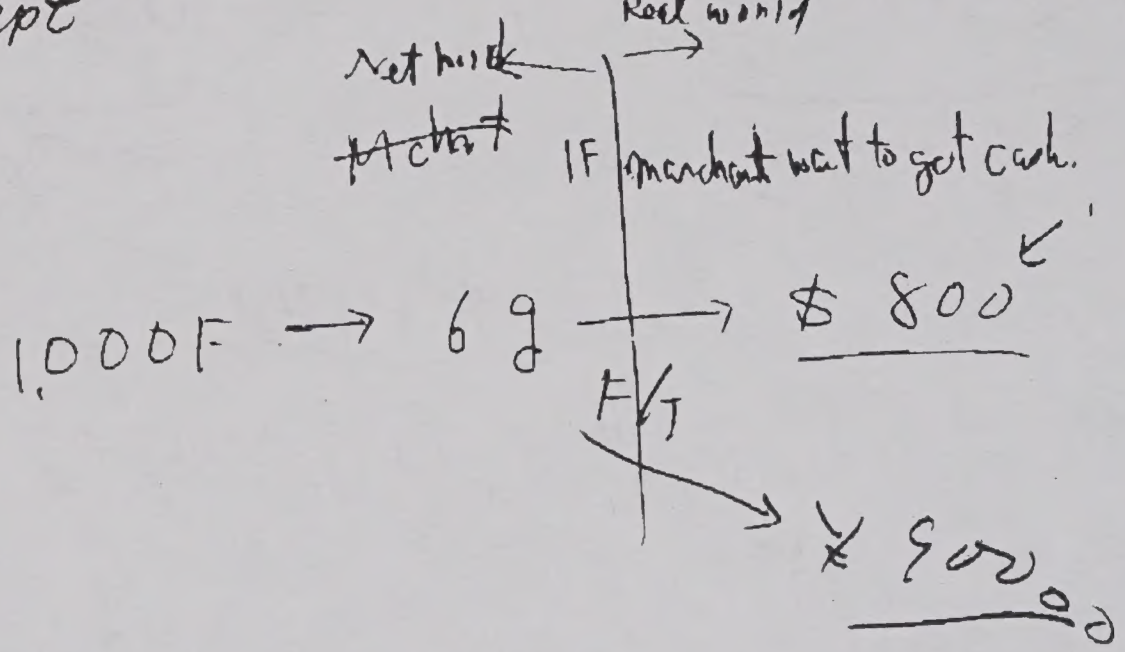
2. Absence of trusted existing fin. institutions to back concept

3. TAXES
TAX REPORTING
WITHHOLDING

BENEFITS

1. (U.S.) TRUSTED PARTY

2. Off-shore - fewer reg's/taxation law


Network
~~Merchant~~
Real world
IF merchant want to get cash.
1,000F
6 g
$ 800
F/T
¥ 800.00

Existing "Cyber" Payment Marketplace

Weaknesses/Gaps

1. Payment vs. transaction
   (absence
2. Potential privacy breach
3. Privacy-centric vs. Market info.-centric

→ 4. Increased liability for accuracy of content

5. Absence of existing public key infrastructure
   (i.e. signing content - authentication)
6. Machine to machine transaction
   (Digital Silk Road)
7. Absence of thorough 'protocol analysis

Strengths

1. DigiCash → exchanges "real" money
2. Aggregation
3. Existing (albeit small) market share/ownership
4. Supported by well-known standards org's. + consortiums (vendors; "SET")

HyperCoin 3

Kai Nishi

Goals for the Meeting

30 November - 1 December

Primary Results Desired for HyperCoin 3

- Determine whether Micropayments are a good business opportunity
- Determine whether Joint Venture Partners are needed
  - Preference is to do without partners for fast response
- General agreement on significant potential for success
- Closure (HyperCoin 3 is the last workshop)

Some Constraints

- Don’t touch browser
- Don’t touch server$_P$
- Provide Mint, Bank, and Archival Services
- Archival Services -- Xanadu

Proposed Strategy

- Start or purchase a bank
  - Names: “First HyperBank”, “First International Bank”, or “First Virtual Bank”
  - What should be the venue for the bank
- Use Gold as a barter medium
- Do it quickly and gain experience in the market
- Must be international

HyperBank Provides Limited Services

- No payments required for settlement
- No payments required for an account
- Exchange services are provided
- No Savings
- No Loans

No investment management, financial services, etc..

How the Bank (Mint) Makes Money

- Conversion of money to gold (scrip) is charged a transaction fee.
- Conversion of gold (scrip) to money is charged a transaction fee.

Alternative Strategies

- Partner with another player
- Explore other markets as more lucrative

Problems & Issues

- Regulatory issues
- Privacy and Anonymous Transactions

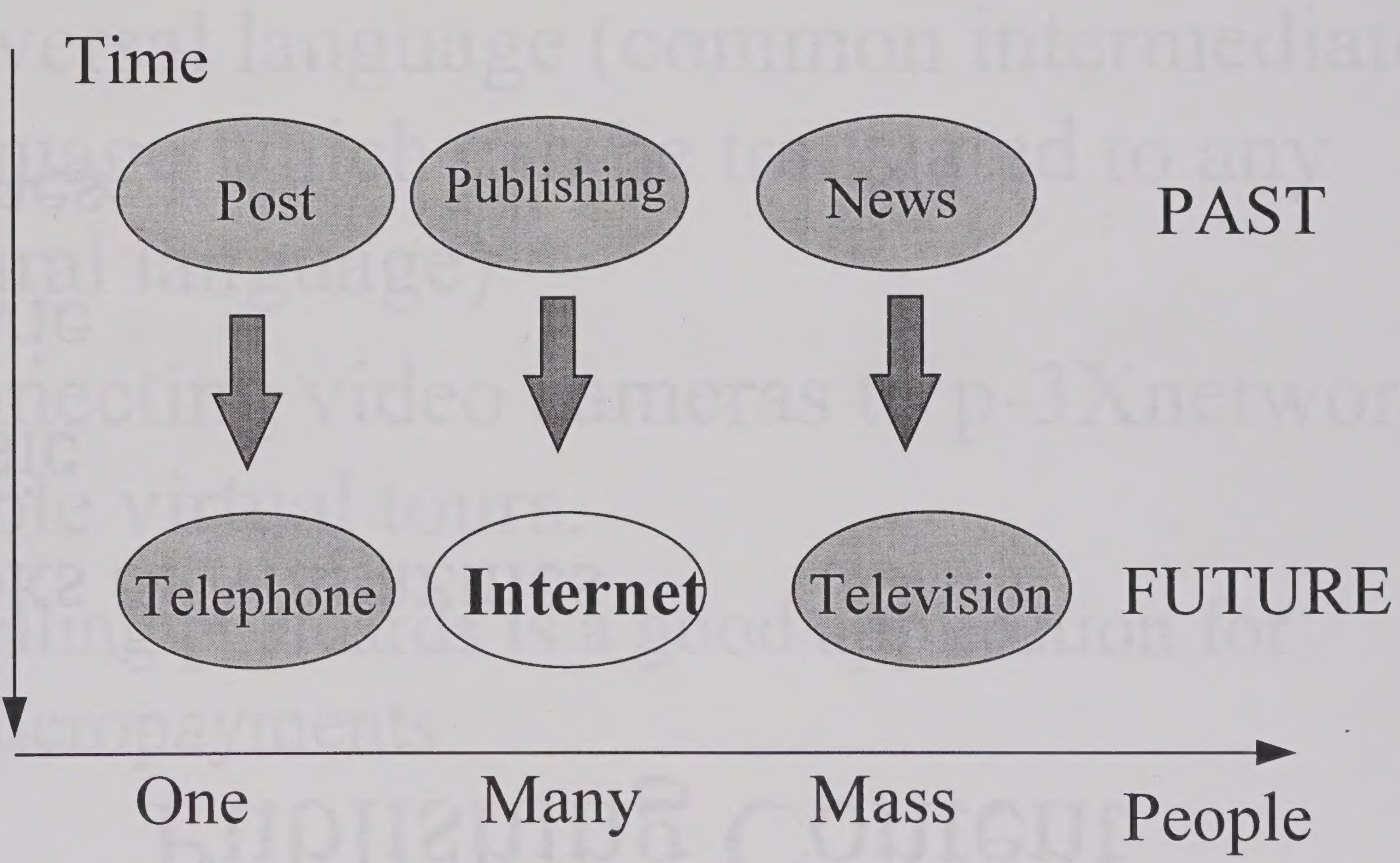
First NanoTransaction Symposium

- Open invitation for attendees and speakers
- Invited attendees as well (?)
- Technologists, Analysts, Market Representatives
- Moderate sized meeting (200)

The Future

- The Internet will be more important than Personal Computing

20 Century Media Map



Time

Post
Publishing
News
PAST

Telephone
Internet
Television
FUTURE

One
Many
Mass
People

Publishing Content

- Books & Magazines
- Music
- Movie
- Games

Other Kay Nishi Projects Which Contribute

- Universal language (common intermediate language which can be translated to any natural language)
- Connecting video cameras to p-3Xnetwork to enable virtual tours.
  - selling postcards is a good application for micropayments

Client-Side Micropayments

Dennis Allison

HyperCoin 3

30 November 1996

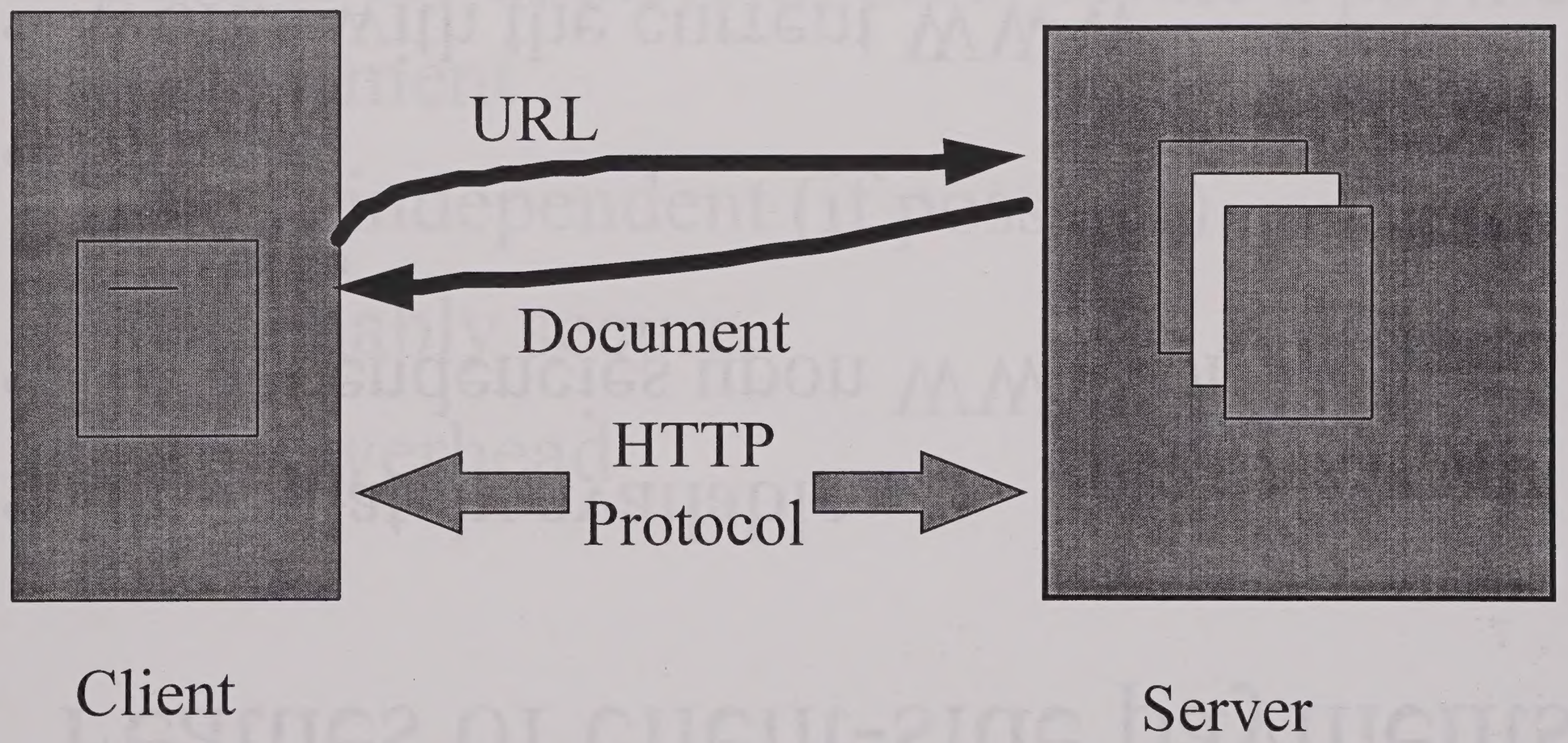
Goals

- Simple for authors to incorporate a payment requirement
- Server independent (if possible)
- Reasonably secure
- Low Overhead

Featues of client-side payments

- Immediately available
- No dependencies upon WWW or browser changes
- Simple
- Works with the current WWW
- Resolves some proxy cache problems

Request response on the WWW



URL

Document

HTTP
Protocol

Client

Server

Java and Javascript can support client-side payments

- Java and a subset of Javascript are supported by Netscape and Microsoft
  - Mosaic, Spyglass, Lynx won’t be able to reference micropayment docs
- These languages seem to provide the features needed
- client references a URL pointing to a payment required document

HTTP 1.0 -- Client

- client contacts server using URL sending method {GET, POST, HEAD}, document address, HTTP version number
- client sends header information, possibly send additional information

HTTP 1.0--Server

- server responds with a status line containing HTTP version, status code, and description
- server sends header information about itself and requested document
- server sends (for a successful request) the data consisting of the document or the result of the server CGI program

HTTP 1.0--Client

- client receives document and closes connection (may be kept open in HTTP 1.1)

What The Bank Provides

- The Bank maintains an account for each possible user. Account support is distributed.
- The Bank provides checklets which can either be a value token or a credit token.

Getting a Trusted Agent running on the Client

- All documents need to reference an Applet which looks for such an agent and, if one is not present, spawns one. All documents for which payment is expected on access must include this Applet. The associated URL is resident at the Bank.

Trusted Agens

- The trusted agent needs to be self-verifying, self-stabilizing to prevent or minimize spoofing. They negotiate with the Bank to obtain the right to spend money on behalf of the client (with appropriate inputs from the client). The trusted agent mediates all document transfers and collects payments from the client. From time to time the agent transfers the accumulated payments to the bank for settlement.

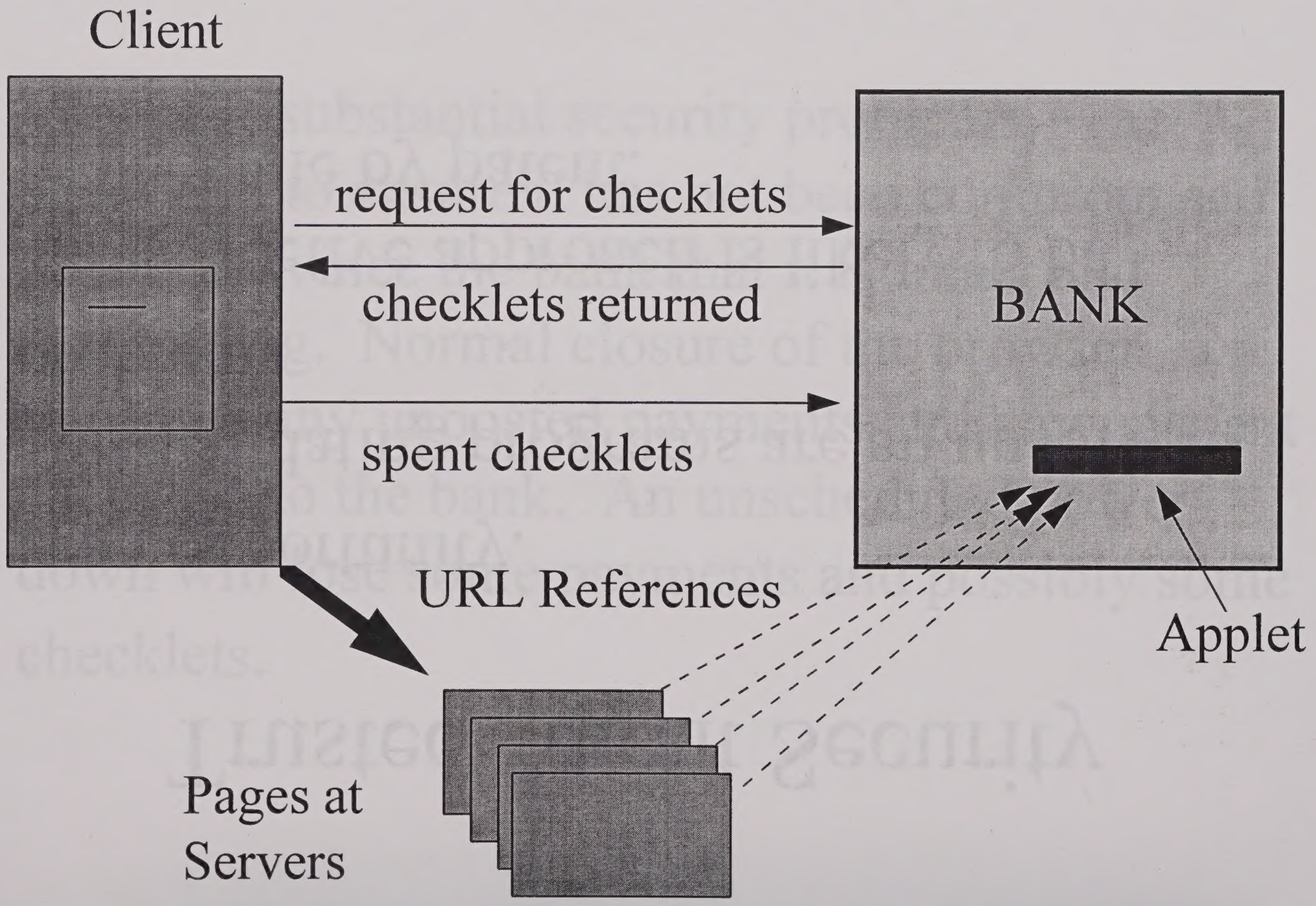
Security of Trusted Agents

- There are substantial security problems. The agent trys to ensure it has not been compromised and to convince the bank that it is intact and functioning. Normal closure of the browser transmits any unposted payments and any unspent checklets to the bank. An unscheduled power down will lose some payments and possibly some checklets.

Trusted Agent Security

- is an opportunity.
- Self-validating programs are an unsolved problem.
- Any effective approach is likely to be protectable by patent.

The Role of the Bank



Client

request for checklets

checklets returned

spent checklets

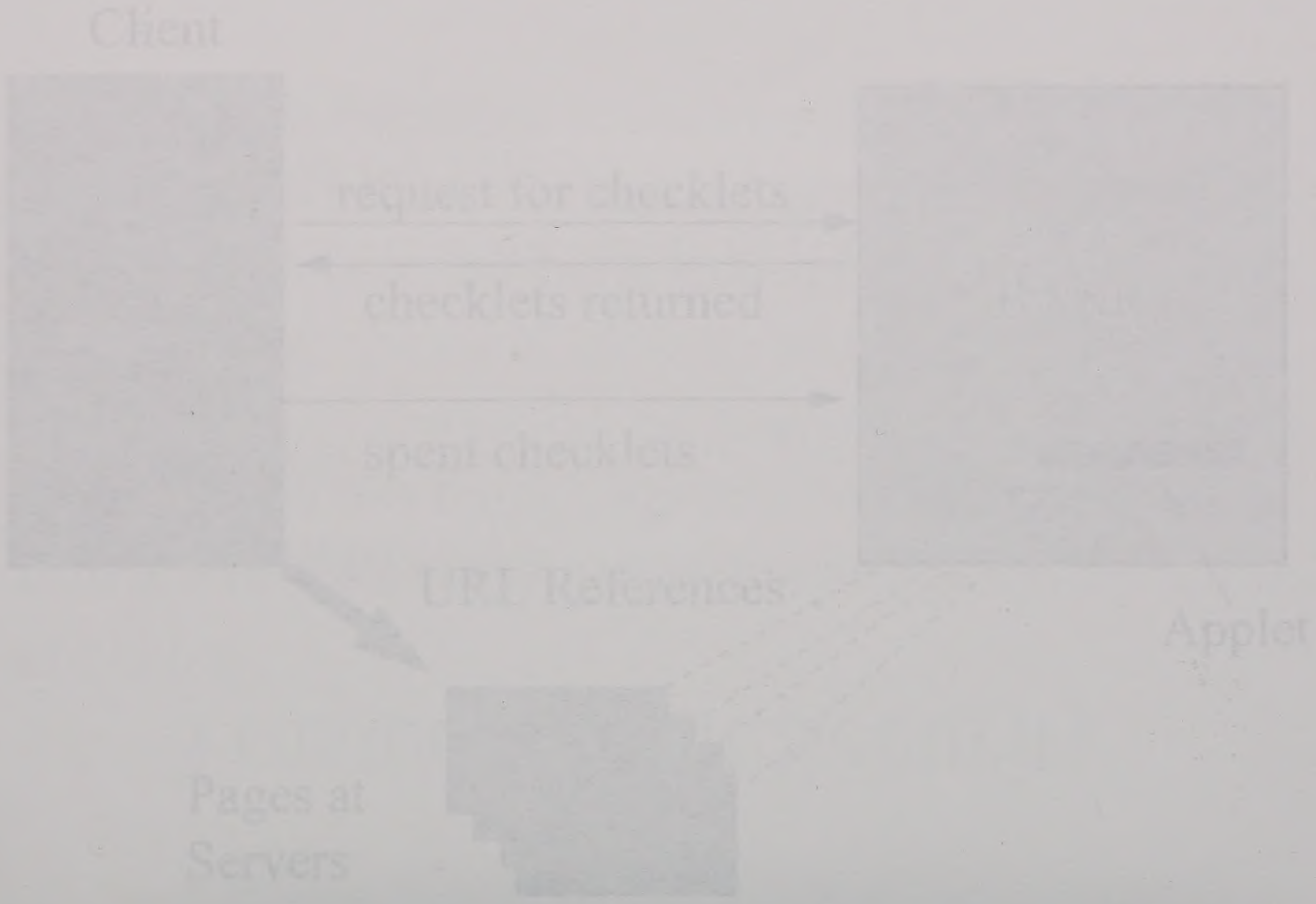
URL References

BANK

Applet

Pages at
Servers

The Role of the Bank



Client

request for checklets

checklets returned

spent checklets

URL References

Applet

Pages at
Servers

HyperCoin 3 Conclusions

Morning Session
November 30 - Dec 1, 1996
San Francisco

Micropayment Systems Are An Opportunity

- Clear market opportunity
- Distinct from publishing opportunity

Workshop/Exploration Phase Is Over

- Technology -- no showstoppers
- Market Exists and is growing
- Time to move forward to one or more real projects

It's Time to Focus on Implementation

- Not an investment
- No cash-out (no venture funding, no buy-out)
- Kay Nishi -- Wants to be first internet publisher

Immediate Goals (Kevin Vaughan)

- Create structure to put product out into the market
  - keep micropayment and publishing component distinct
  - keep goals to things which can be attained
  - no substantial research component

Goal -- Deploy Something Quickly

- Micropayment System? Publishing System? Xanadu
- Need to create Framework
- Need to develop infrastructure with partners

Goals for Sunday (1 Dec)

- Limits and constraints for “better micropayment scheme”
- Limits and constrainst for “better publishing scheme”
- Define something which can be deployed within 12 months

See slides from sessions

Electronic Publishing

1. Custom Books (students) (DocuTech - Xerox)
2. Citation index
3. ARCHIVAL JOURNALS (Reviewed) → indexing issue
4. MONOGRAPHS (NOT Reviewed)
5. Book (2-3 yrs.) NOT GOOD FOR CURRENT RESULTS - "ACADEMIC" PUBLISHING

MIXED MEDIA - NEWSGROUPS, EMAIL, ETC, WEB PAPERS,

DIGITAL LIBRARIES (UNIVERSITIES)

ASCII Publications

Micro PAYMENT

1. Independent creators/artists (requires robust distribution & marketing effort/service/expertise).

   1a. Applet writers/authors (futuristic to some degree)

2. Subscription Services ("MS Electronic Library")

   ISSUES
   - A. Prohibitive Sub cost
   - B. Prohibitive Adv. cost
   - C. Pay-per-view

3. "Token" payments (supported via credit card infrastructure, automatic transfrs,

   Allows R&D (public & private). Browser independent,

   Removes trust issue req'd for funds holding

   trans confirm/minimum → entry fee

HyperCoin3

Sunday Afternoon
Nov 30 - Dec 1, 1996

Adjunct Players

- state lotteries, etc
- pay per view
- on-line gambling
- coin op games
- magazines, books, newspapers
- pgoto stock
- applets
- software
- information
- fortune telling
- utility sensors reports
- stocks and bonds

Adjunct Players

- purchase goods
- purchase services
- transactions
- reservations

- access to markets (pay for attention)

Who can use the product/service?

Buyers

- anyone who uses web
- agents looking for content

Sellers

- advertisers
- publishers (books, mags, music, pictures)

Who can use the product/service?

Buyers

- anyone who uses web
- agents looking for content

Sellers

- advertisers
- publishers (books, mags, music, pictures)

What NOT to do (1)

- Don’t require tracability (overhead)
- Do not require additional computerized software components for client.
- Design for the future, match the present, utilize and exploit the past
- Do not design a static money system; all current monetary systems are dynamic and based on trust

What to do (1)

- Define a micro-transaction market, market driven
- define a marketplace
- Contain cost of privacy relative to what the market will bear
- scalable system
- distributable system.

What to do (2)

- use current tools, browsers, and servers. Must be compliant with currently accepted web protocols
- Provide incentives for use for both client and server
- Provide clearly stated process for disputes (rescue rather than refund)

What to do (3)

- Design for trust and credibility

What to do (3)

- Limit international exchange exposure
- Expose legal exposures due to international law issues
- Rapid prototype deployment, rapid develoment

Incent Participants vis a vis other Payment schemes

- card systems
- banking rules
- &tc

What is marketplace?

- characteristics <=> marketplace

Who can use the product/service?

- anyone who uses the web
- active agents (programmers)
- businesses and individuals are similar

Authentication

- Merchants (Tax ID)
- Individuals (self-reporting, Tax ID)

Legal Issues

- Tax issues for income
- International Currency
-

Token MicroPayment System

Benefits

Allows entry
Provides live R+D real-time
Removes trust issue for funds holding
everages existing credit card & bank infrastructure
Browser independent
Allows internal R+D
Automatic Account Transfer

Issues / Concerns

- TRUST
- Match to market culture
- Appeal to market wishes/needs (Market perception – trust issue)
- How to handle settlement issues

"credit card companies cannot be trusted"
KS

1. What not to do

2. Define marketplace (product, participants, method/process)
   - products/services Characteristics
   - scale
   - non - face-to-face maybe non-person-person
   - appropriate trust level

What NOT TO DO

1 [illegible]
(no [illegible])

2 Do NOT [illegible]
customized software components
(for client)

3

WHAT TO DO

1 Will define a pure transaction
2 [illegible] market driven mechanism
2 define marketplace
3 [illegible] cost of Privacy relative to what market will bear
4 Scaleable
5 Distributable
6 Must be compliant w/currently accepted web protocols (ex. http)
7 Provide process for dispute [illegible]

What NOT TO DO

1. DON'T REQUIRE TRACEABILITY (overhead)
2. DO NOT REQUIRE Add'l. customized software components (FOR client)
3.

WHAT TO DO

1. Well define a micro transaction
2. create only market driven mechanism
2. define marketplace
3. Contain Cost of Privacy relative to what market will bear
4. Scaleable
5. Distributable
6. Must be compliant w/currently accepted web protocols (ex. http)
7. PROVIDE PROCESS FOR DISPUTE
   - CLEARLY STATED

What to Do   NOT TO DO

. INCENT PARTICIPANTS
vis-à-vis other payment systems,
card systems, banking rules, etc.

. Limit international exchange
exposure
0. Exclude money laundering within
mechanism model
1. Examine legal exposure (tax & otherwise)
2. Rapid prototype + product development time frames

Regulatory + legal issues

What is the marketplace?

(characteristics = requirements)

RCHASE/SHIPPING
tilities
eservations (restaurants)
emote video
strology
Research
DB queries
FORTUNE TELLING, ETC.
usic Providers
SCRIPTIONS

Advertizing
Bookpublishers
Pay-per-view/V.O.D.
Coin-op-gaming
Online gambling
Magazines
Newspapers
Telecommunications
Photostock/Clip Art
Applets
Other software Components

Qualitative/Quantitative Marketers

DIRECT MARKET ACCESS

Buyers

anyone who uses the web
(business + individuals)

Sellers

Merchants (tax i.d
Individuals (self report below legal li
(Buy, sell, withraw)

Summary

1. Product/Service/Market Characteristics
2. Customers/Sellers Identified
3. Case Examples for businesses

4. Next steps identified:
    - a) Business Plan Creation (entity description, geography, divisional structure, ownership, board, lab, etc. ...)
    - b.) RESOURCE IDENTIFICATION
    - c.) R&D long-term (12-month) objectives
    - d.) 1.0 Product deployment target schedule
    - e.) Research component identified

micro payment Market definition

- NON face-to-face transactions

- encompasses existing purchase & deliver markets (mail order)

- supports previously unsupported business transactions through availability of particlized currency via accounting and aggregation process

- expands electronic commerce to businesses not currently exploiting it (e.g. utilities, pay-per-view, etc.)

HyperCoin3

Summary

Nov 30 - Dec 1, 1996

HyperCoin3

Summary

Nov 30 - Dec 1, 1996

Market/Product/Service Characteristics

Micropayment Market Definition

- Non-face-to-face transactions
- Encompasses existing purchase and deliver market model (mail order)
- Supports previously unsupported business transactions through availability of particularized currency via aggregation accounting process
- Expands electronic commerce to businesses not currently exploiting it (for example, utilities, pay-per-view, etc.)

Customers/Sellers Identified

Who will use micropayments

- Anyone who uses the web
- Some will use it to buy goods and services
- Some will supply goods and services
- Some will do both
- active agents searching for things on the web

Authentication/Scoring Matrix

- Individuals -- tax ID? self-reporting
- Businesses -- tax ID
- Volume dependent authentication

Who can use the product/service?

- **Buyers**
- anyone who uses web
- agents looking for content

Sellers

- advertisers
- publishers (books, magazines, music, pictures)

Case Examples for Businesses

Target Businesses (1)

- state lotteries, etc.
- pay per view
- on-line gambling
- coin op games
- magazines, books, newspapers
- photo stock
- applets
- software
- information
- fortune telling
- utility sensors reports
- stocks and bonds

Target Businesses (2)

- purchase goods
- purchase services
- transactions
- reservations
- access to markets (pay for attention)

Guidelines

What to do (1)

- Define a micro-transaction market, market driven
- define a marketplace
- Contain cost of privacy relative to what the market will bear
- scaleable system
- distributable system.

What to do (2)

- use current tools, browsers, and servers. Must be compliant with currently accepted web protocols
- Provide incentives for use for both client and server
- Provide clearly stated process for disputes (rescue rather than refund)

What to do (3)

- Design for trust and credibility
- Limit international exchange exposure
- Expose legal exposures due to international law issues
- Rapid prototype deployment, rapid development
- Incent Participants vis a vis other Payment schemes (cards, banking rules, etc.)

What NOT to do (1)

- Don’t require tractability (overhead)
- Do not require additional computerized software components for client.
- Design for the future, match the present, utilize and exploit the past
- Do not design a static money system; all current monetary systems are dynamic and based on trust
- banking rules
- etc.

Legal Issues

- Tax issues for income
- International Currency

Next Steps Identified (1)

- Business Plan Creation
    - entity description
    - geography
    - divisional structure
    - ownership
    - board
- Resource Identification (including staffing)

Next Steps Identified (2)

- Research and Develpment Objectives
- 1.0 Product Development Target Schedule
- Research Component Identified

MicroMoney ‘97

- A Symposium of
  - Micropayment vendors
  - Potential content providers
  - Analysts
  - Economists
  - Protocol and crypto specialists

Time and Schedule

- Early May
- 2 Day Conference
- Either Mon-Tues or Thurs-Fri

Staffing

- D. Allison, Chair
- 10 member Program Committee
- Commercial Organizer doing logistics

Focus

- Academic Persoective on Micropayments/Transactions
- Market Comparisons

Scale

- Small (200-400 people)
- But what if it grows....

Involve Attendees in Payment Demos

- register on-line
- pay by some micropayment scheme

Demos of Hypermint and Competition

- Could be part of presentations
- Needs to be more than a demo -- needs some explanation of what’s really going on

Exhibits & Tutorials

- Should there be commercial exhibits sold by the booth?
- Should there be tutorials for webmasters and ISPs?

Tentative Program

- Protocol Technology
- How to Embed in Web Docs
- Privacy (Lori Feena, EFF)
- Company Presentations
- Comparative Papers

Possible Keynoters and Plenary Speakers

- Alan Greenspan
- Bill Gates
- Michael Millikan
- SFI speaker
- Lloyd Benson (Treasury Secretary)
- Bank CEOs or the like
- Paul Saffo
- Colin Crook
- Ted Nelson
- John Gage
- Jock Gill
- Panel with Auditing Firm (Big 5)
- <International Presenters>

What Venture Community Participation?

- as evaluators
- as attendees

Micropayment Vendors

- Working Systems
- Non-Working Systems

Progam Committee

- Dennis Allison
- Tsutomu Shimomura
- Kevin Vaughn
- Brian Reid
- Whit Diffie
- Jock Gill

- <Who from Japan?>
- <Who from Europe?>
- <Who from Pacific Rim?>
- <Who from Middle East?>
- plus others....

Schedule

- Program Committee -- Dec 14
- Location and Date -- Dec 14
- Program Committee Meeting -- Early Jan
- Call For Papers (mostly invited) Mid Jan
- Paper Review -- Late Feb
- Acceptance -- Early April
- Conference -- Early May

Symposium Details — State of the Art SYMPOSIUM

Chair: Dennis A.

~~April~~ May – 1st week – Location: S.F. or S.J.

MICRO MONEY '97

A Symposium on MicroMarkets
& MicroCommerce
& MicroTransactions

Target: Micropayment vendors, potential content providers

Focus: Academic Perspective on MicroPayments/Transactions
Comparitive of market

Scale: Small

Demo of Hypermint & Competition

Exhibits

Tentative Program

Techapps\Internet, Networking, Network Applications & Security\Network Payment Systems

Network Payment Systems and Digital Cash

2 Day Tutorial - 3-4 December 1996

This tutorial will enable you to...

- Identify the potential for electronic trading across networks and the inherent risks involved
- Understand the security mechanisms involved in payment systems
- Compare and evaluate the different types of payment system that are available.
- Select appropriate payment systems for particular application areas
- Gain an understanding of the significant industry alliances driving the major payment system alternatives.
- Understand the need for Micropayment Systems for low-value information
- Appreciate where payment systems are headed in the future

Click here for first step to Web site devoted to Network Payment Mechanisms and Digital Cash

---

Scope and Purpose of Tutorial

The tutorial examines all the key technologies, including essential security techniques, for network payment and digital cash systems, and assesses their strengths and weaknesses and suitability for different business applications.

It not only provides this important technical foundation but also describes and compares the products and services that implement the key technologies, and how they are fairing in the market place. It also examines the shape of the emerging network payment and digital cash "industry" in terms of the leading players.

Convenience, security, privacy and cost drive the market

There are numerous conventional payment methods--cash, cheques, charge cards, credit cards, debit cards, travellers cheques, tokens, and a variety of banking instruments--each with its on advantages and disadvantages, in terms of convenience, security, privacy and cost, for customer, seller and financial organisation.

Digital money is likely to have as much diversity, perhaps even more, as the market goes through a period of development and testing and we discover and exploit new network business applications.

Internet and Web raise the stakes

The Internet and World Wide Web have dramatically boosted interest in network payment systems and digital cash and increased their significance. The technical challenges are tougher because payment systems must now work over worldwide open "best-effort" networks and serve new kinds of businesses and payment models made possible by such universal network connectivity. The opportunities have

attracted a wide range of players, from big and established organisations such as Mastercard, Visa, Microsoft, and the major banks, to newcomers such as CyberCash, DigiCash, First Virtual, and Millicent, to name only a few. Secure bank card systems suitable for the Internet are coming into place but they are unlikely to meet all requirements. There will be more than one way of paying for something on the Internet. The Internet and World Wide Web are the special focus of this tutorial.

Micropayment systems will enable new kinds of business

The Internet and the World Wide Web have the potential to create vast new markets for information services and "knowledge" and software products but the full potential will only be realised through payment systems that economically deal with very small amounts. They must also allow all buyers and sellers to participate whether or not they belong to a centrally administered merchant system. The tutorial examines leading micropayment technologies and systems.

Privacy and anonymity - the consequences of technology

Some forms of electronic money make it possible to completely monitor the details and history of an individual's expenditure and pose a threat to individual privacy on a scale never seen before. It is too soon to predict what consumers will accept in the market (and what some governments will allow) but the tutorial shows how various technologies contribute to the problem or solve it.

What companies should be interested in network payment systems

Any company or organisation seeking to genarate business over networks especially the Internet or via the Web must examine the wide range of network payment mechanisms and digital cash systems now becoming available and choose those methods and service providers that give the greatest potential for network generated business growth. All banks and financial institutions face a time of upheaval and they need to understand and harness these new technologies before they find their positions undermined by rapid market developments. Network payment systems bring opportunities for systems and software houses who serve companies using the Web or who serve the financial sector, and for suppliers of network technology, browsers, software agents, MIME-based messaging, EDI and electronic commerce systems. New services and new products such as electronic "wallets" will also be required.

Who is the tutorial for

The tutorial is for IT, MIS and networking executives whose companies are using or plan to use networks, especially the Internet and the Web, to generate business and serve their customers. In particular it will be of value to software developers and managers in a wide range of business sectors and banking and finance, concerned with electronic commerce .

Outline Contents

- **Motivation for Electronic Payment**
    - Potential for Electronic Commerce
    - The World Wide Web as a Commerce Enabler
    - Generic Web Payment Example
    - Examples of Payment Systems in Operation

  - Examples of Payment Systems in Operation
  - Size of the Marketplace

- **Characteristics of Conventional Payment Systems**
  - Payment Systems Models
  - Conventional Payment Instruments:
    - Cash, Cheque, Draft, Credit Card, Funds Transfer etc
  - Clearing Mechanisms
  - Payment Statistics

- **Introduction to the Cryptographic Techniques Used in Payment Systems**
  - The security risks associated with payment
  - Symmetric Encryption
    - DES, Triple DES, IDEA
  - Asymmetric Encryption
    - RSA
  - Message Digests
    - Signatures, Blind Signatures
  - Certificates
  - Certification Authorities
    - The Need to Establish a Chain of Trust
    - Typical CA hierarchy for Payment

- **Electronic Payment Systems Characteristics**
  - - On/Off-line
  - - Hardware/Software based
  - - Anonymity/Audit Trail
  - - Macro/micro Payment
  - - Information/Hard Goods/ Services

- **Proposed Payment Systems**

  This section of the course will cover a number of the most influential and important payment systems that are being proposed or are in operation. Given the rapid changes taking placethe following list should be taken as a rough guide to what will be covered.

- **MacroPayment Systems**
  - Netscape Commerce Server/SSL
  - VISA/Mastercard Secure Electronic Transactions (SET)
  - First Virtual
  - NetBill
  - CyberCash
  - E-cash
  - Netcash
  - Mondex

- **Micropayment Systems - Payment for WWW Information Resources**
  - Millicent
  - Micromint
  - WWW Consortium Micropayment Transfer Protocol (MPTP)

- **Applications of Payment Systems**
    - Software Distribution with Ziplock
    - Versatile Virtual Vending

- **Integrating Payment Systems with the World Wide Web (WWW)**
    - WWW Client/Server Independence/Dependence
    - CGI vs Java/Javascript

- **Regulation of Electronic Payment Systems - Concerns of Central Banks**

- **Current Status and Future Prospects for Electronic Payment**

The Presenters

Dr. Donal O'Mahony is a lecturer in the Computer Science Department at Trinity College Dublin, where he co-ordinates a research group working on a variety of projects in networking, telecommunications and data security. He is co-author of Local Area Networks and their Applications published in English by Prentice-Hall and in Japanese by Kaibundo. He has been active in the fields of networked security services, Electronic Data Interchange(EDI) and Electronic Commerce for more than 10 years and has published and lectured widely in these areas. More recently, he has supervised a number of innovative research projects in Teleshopping, Electronic Commerce and Payment systems.

Michael Peirce graduated with a B.A.(Mod) in Computer Science from Trinity College, Dublin. His final year dissertation concerned the design of a new scalable anonymous electronic payment mechanism for the purchase of goods and services on the WWW. The results of this work were published in the 4th International WWW conference in December 1995. He has acted as the maintainer of a highly popular Web site on Network Payment Mechanisms and Digital Cash since December '94. He was senior technical reviewer for the popular *Internet: The Complete Reference*, and chief researcher for *The Internet Yellow Pages*, both published by Osborne/McGraw-Hill. Currently he is working as a researcher in the area of mobile communications, with the Networks and Telecommunications Research Group (NTRG) at Trinity College, Dublin

.

Return to Home Page | **Send for further details**

Technology Appraisals Ltd
webmaster@techapps.co.uk
Phone +44 (0)181 893 3986
Fax +44 (0) 181 744 1149
82 Hampton Road, Twickenham
TW2 5QS UK

©Technology Appraisals Ltd 1996